jamk.fi

RFID-tunnistetietojen tallennus tietokantaan

Jouni Ylönen

Opinnäytetyö
Marraskuu 2015

Tietoverkkotekniikan koulutusohjelma
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Ylönen, Jouni	Opinnäytetyö	6.11.2015
	Sivumäärä 51 + 25	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: (x)

Työn nimi
RFID-tunnistetietojen tallennus tietokantaan

Koulutusohjelma
Tietotekniikan koulutusohjelma

Työn ohjaaja(t)
Antti Häkkinen
Mika Rantonen

Toimeksiantaja(t)
Prospectum Oy
Pekka Neuvonen

Tiivistelmä

Opinnäytetyön toimeksiantajana toimi Prospectum Oy. Toimeksiantajalla oli tarve kävijäystävällisestä langattomasta tunnistusjärjestelmästä. Opinnäytetyön tavoitteena oli tutkia erilaisia langattomia tekniikoita ja kehittää langatonta tekniikkaa hyödyntävä tunnistautumisjärjestelmä kävijätunnistukseen tapahtumakäyttöön.

Aluksi työssä tutkittiin ja vertailtiin langattomia tekniikoita ja niiden tarjoamia ominaisuuksia. Tutkituista tekniikoista valittiin toimeksiantajan kanssa yksi, jota hyödyntäen tekninen toteutus tehtiin. Työssä perehdyttiin myös tietokantarajapintoihin sekä verkkosivujen rakenteisiin ja toiminteisiin. Toteutukseen valittu RFID-tekniikka ja sillä kerätty lukuprosessin data siirrettiin MySQL-tietokantaan ja tietokannasta verkkosivustolle.

Vaatimusmäärittelyt työlle olivat tunnistuksen vaivattomuus ja järjestelmän nopeuteen sekä kapasiteettiin liittyvät kysymykset. Työn toteutuksella pystyttiin vastaamaan sille esitettyihin kysymyksiin. Opinnäytetyön lopputulos oli toimiva RFID-tunnistusjärjestelmä, jolla pystyttiin todistamaan tekniikan soveltuvuus vaadittuun käyttötarkoitukseen. Työn tulos antaa toimeksiantajalle suunnan tämänkaltaisen järjestelmän tuotteistukseen.

Avainsanat (asiasanat)
RFID, NFC, iBeacon, MySQL, PHP, Node.js, tietokanta

Muut tiedot
-

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Description

Author(s)	Type of publication	Date
Ylönen, Jouni	Bachelor's thesis	6.11.2015
		Language of publication: Finnish
	Number of pages 51 + 25	Permission for web publication: (x)

Title of publication
Saving RFID tag information to database

Degree programme
Information Technology

Tutor(s)
Antti Häkkinen
Mika Rantonen

Assigned by
Prospectum Oy
Pekka Neuvonen

Abstract

This bachelor's thesis was assigned by Prospectum Oy, a company which had a need for a user-friendly wireless identification system. The thesis aims to study different wireless technologies and develop a wireless visitor identification system for event use.

At first, the wireless technologies and the features they offered were studied. One of the technologies studied was chosen for the technical implementation. The thesis also presents database interfaces and the structures of webpages. RFID was chosen for the technical implementation of the thesis and the data acquired was sent to a MySQL database and off to the management website.

The requirement specifications of the thesis were user-friendliness of the identification process and the questions regarding both the speed and the capacity of the whole system. The technical implementation was able to answer these questions. The end result of the thesis was a fully functional RFID identification system capable of proving the suitability of the technology for the needed application.

The end result of the thesis gives guidelines for the assigner to produce this kind of an identification system.

Keywords/tags (subjects)
RFID, NFC, iBeacon, MySQL, PHP, Node.js, database

Miscellaneous
-

Sisältö

Kuviot

Lyhenneluettelo

Apache Verkkosivustoja ylläpitävä palvelu
ANSI American National Standards Institute, standardointiorganisaatio
ASCII American Standard Code for Information Interchange, tietokoneiden merkistö
CLI Command Line Interface, käyttöliittymä, yleensä hallitaan tekstikomennoilla
DATETIME tietuetyyppi, päivämäärä ja kellonaika
DBMS Database Management System, tietokannan hallintasovellus
HF High Frequency, aaltotyyppi
HTML HyperText Markup Language, Internet-sivujen kieli
HTTP HyperText Transfer Protocol, selaimen ja palvelimen välillä tiedonsiirtoprotokolla
INT Integer, tietuetyyppi numeroille
ISM Industrial, scientific and medical radio bands, taajuusalue, jonka käyttöön ei vaadita lupaa
JavaScript ohjelmointikieli
LBT Listen-Before-Talk, kuunteluantenni
LF Low Frequency, aaltotyyppi
MAC Media Access Control, rautaosoite, laitteen rajapinnan fyysinen osoite
Microwave Mikroaalto, aaltotyyppi
MySQL Tietokantasovellus
MySQLi PHP:n laajennus, mahdollistaa tietokantaan yhdistämisen PHP:lla
Parser Jäsennin, ohjelma koodin merkkijonon oikeellisuuden tarkastukseen
PHP HyperText Preprocessor, web-palveluissa käytetty ohjelmointikieli
RF Radio Frequency, radiotaajuus
RFID Radio Frequency Identification, Automaattinen tunnistustekniikka
RST Reader Startup Tool, Sirit 510 -lukijan virallinen hallintaohjelmisto
RX Receive, vastaanottaa, antennin ominaisuus
SQL Structured Query Language, kyselykieli
Socket soketti, rajapinta eri protokollien välillä
Tag RFID-tunniste, kiinnitetään luettavaan kohteeseen
TX Transmit, lähettää, antennin ominaisuus
UHF Ultra High Frequency, aaltotyyppi
VARCHAR Variable Character Field, tietuetyyppi numeroille ja kirjaimille

1 Lähtökohdat opinnäytetyölle

1.1 Prospectum Oy

Prospectum Oy tarjoaa ratkaisuja yritysten ja yhteisöjen tapahtumanhallintaan, osallistujaviestintään ja osallistamiseen. Yritys aloitti toimintansa vuonna 2010. Yrityksen erikoisosaamista on monikanavaisen osallistumiskokemuksen kehittäminen niin tapahtumissa, työyhteisöissä kuin kouluissakin. Yritys haluaa tarjota asiakkailleen innovatiivisia ratkaisuja viestinnän ja vuorovaikutuksen tarpeisiin. (Prospectum Oy. n.d.)

Yritys on kehittänyt osallistumiskokemusta vuorovaikutuksen avulla lukuisissa erilaisissa tapahtumissa, oppilaitoksissa sekä yrityksissä ja organisaatioissa. Osallistumiskokemus on yrityksen mielestä yksi tärkeimmistä sitoutumiseen ja yksilön viihtyvyyteen vaikuttavista asioista. Yritys tavoittelee sitä, että osallistuminen on helppoa ja jokaisella osallistujalla tulee olla tasa-arvoinen osallistumismahdollisuus paikasta ja päätelaitteesta riippumatta. Palveluissaan yritys panostaa erityisesti helppokäyttöisyyteen ja tasa-arvoisen osallistumisen mahdollistamiseen. Yritys on vakiinnuttanut mobiiliosallistumisen mahdollistavan Viestiseinä®-palvelun osaksi suomalaisia tapahtumia. (Prospectum Oy. n.d.)

Prospectum Oy:n toimistot sijaitsevat Jyväskylässä ja Helsingissä. Kumppaniverkoston kautta he palvelevat asiakkaitaan myös valtakunnallisesti. Yrityksen tavoitteena on palvella asiakkaita nopeasti ja joustavasti. Asiakkaina yrityksellä on merkittäviä suomalaisia organisaatioita. Lähivuosien aikana yrityksen tavoitteena on palvella myös ulkomaisia asiakkaita. (Prospectum Oy. n.d.)

1.2 Lähtökohdat

Yritys oli pohtinut uutta palvelua, jonka avulla pystyisi tunnistamaan kävijöitä ja kävijämääriä tapahtuma-alueella. Yrityksellä ei kuitenkaan ollut resursseja eikä aikaa kehittää palvelua käytännön tasolla. Yrityksen jo olemassa olevaan Viestiseinä®-järjestelmään pitäisi pystyä liittämään moduulina uusi järjestelmä, joskin tarvittaessa muokattuna. Viestiseinä®-palvelu kerää tietoa Internetin syötteistä, esimerkiksi

Twitter ja Facebook. Viestiseinä®-palvelulla kerätyt syötteet on mahdollista näyttää tapahtuman yleisölle videotykin avulla.

Toimeksiantaja tahtoi tietää, miten tietoa saisi kerättyä tapahtuman kävijöistä langattomasti ja vaivattomasti. Millainen järjestelmä siihen tarvittaisiin? Sopivasta tekniikasta ei ollut juurikaan taustatietoja. Asiakaskokemuksen tulisi kuitenkin olla mahdollisimman helppo, ja tapahtuman kävijöiltä ei tarvittaisi toimenpiteitä tiedonkeruuseen liittyen; ”kävijä astelisi porttien läpi tapahtumapaikalle, ja hänen saapumisensa pystyttäisiin näkemään tietokannasta”.

1.3 Työn tavoitteet

Työn tavoitteena oli vertailla erilaisia langattomia tekniikoita ja niiden ominaisuuksia. Esimerkkejä ominaisuuksista ovat laitteiston hinta, koko ja ohjelmoitavuus. Vertailun perusteella tuli valita sopivin tekniikka yrityksen tarpeisiin nähden. Valitun tekniikan ympärille piti toteuttaa langaton järjestelmä, jolla pystyttäisiin esittelemään datan keräämistä tapahtuman kävijöistä heidän saapuessaan tapahtumapaikalle. Yritys ei suoraan ota käyttöön opinnäytetyön tulosta sellaisenaan vaan mahdollisesti osia siitä. Työn päätavoite on selvittää, että tällaisen järjestelmän luominen on mahdollista valitulla tekniikalla ja laitteistolla. Laitteistosta ja lukuprosessista haluttiin mittaustietoa, jotta toimeksiantaja saisi käsityksen langattoman tunnistusjärjestelmän resursseista ja rajoituksista.

Työ toteutettiin suunnittelemalla hallintapaneeli verkkosivupohjalle, jonka avulla RFID-lukijan ja tietokannan hallinta onnistuisi. RFID-lukijaa ohjattaisiin skriptin avulla. Tietokantaan tallennettuja RFID-tunnisteita tulisi olla mahdollista tarkastella hallintasivun kautta myöhemmin. Toimeksiantaja toivoi hallintasivun ja tietokannan välille PHP-rajapintaa.

2 Teoria

2.1 Langaton tiedonsiirto

Langattomalla tiedonsiirrolla voidaan tarkoittaa tiedonsiirtoa käyttäen apuna radiotekniikkaa sekä valoa. Tarkemmin eriteltynä langattomia tekniikoita ovat esimerkiksi radioaallot ja radioliikenne, mikroaallot, infrapuna sekä laser. Luultavasti tunnetuin radiotekniikka on nykypäivänä Wi-Fi eli 802.11-tekniikka, jota käytetään esimerkiksi kodeissa, toimistoissa sekä kauppaliikkeissä ja monissa muissa julkisissa tiloissa.

2.2 RFID

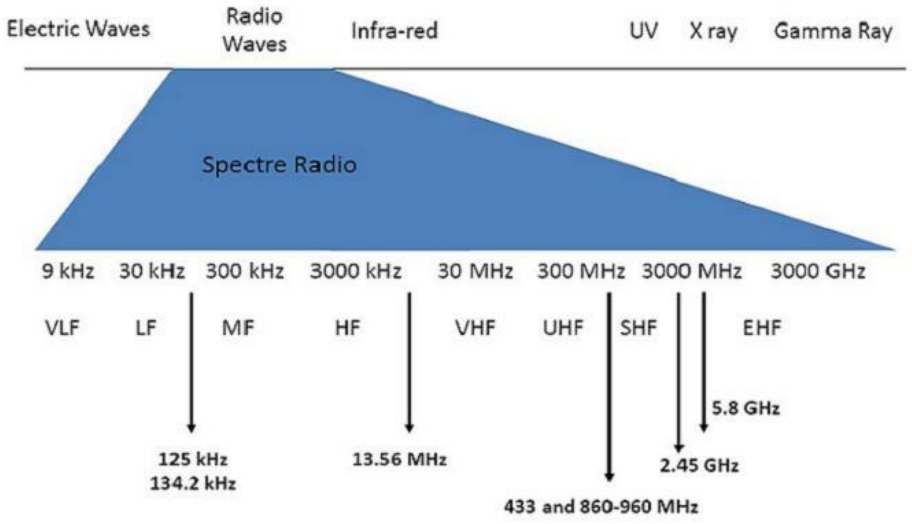
RFID (Radio Frequency Identification) on automaattinen tunnistusmenetelmä, joka hyödyntää radioaaltoja tiedonsiirrossa. RFID-järjestelmä koostuu RFID-tunnisteista eli tageista, RFID-lukijasta antenneineen, hallintatietokoneesta ja hallintasovelluksesta. RFID-järjestelmä toimii niin, että RFID-tunniste kiinnitetään haluttuun merkattavaan kohteeseen. Kiinnitetty tunniste voidaan lukea radioaaltoja käyttäen, kun tunniste saapuu lukijan kenttään. Tunnisteen sisältämä tieto voidaan siirtää lukijasta hallintasovellukselle jatkotoimenpiteitä varten, esimerkiksi tietokantaan lisäystä varten. Tunniste voidaan myös uudelleenkirjoittaa toiseen käyttötarkoitukseen uudella sisällöllä, toisin kuin esimerkiksi viivakoodi, jota ei pysty uudelleenkirjoittamaan. (RFID FAQ. n.d.)

Taajuus ja tunnisteet

RFID-järjestelmä voi toimia useilla eri taajuuksilla. Lukijan käyttämän radiotaajuuden perusteella RFID-tunnisteet voidaan jakaa matalan taajuuden LF (Low Frequency), korkean taajuuden HF (High Frequency), UHF-taajuuden (Ultra High Frequency) ja mikroaaltotaajuuden tunnisteisiin. Eri taajuuksilla on erilaiset kantamat ja ominaisuudet. Taajuudet ja niiden tyypit on esitetty taulukossa 1. Eri taajuuksien sijoittumisen radioaaltojen spektrissä voi nähdä kuviosta 1. (Violino. 2005.)

Taulukko 1. RFID-tekniikan taajuudet ja ominaisuudet. (RFID frequency ranges. n.d.) (Violino. 2005.)

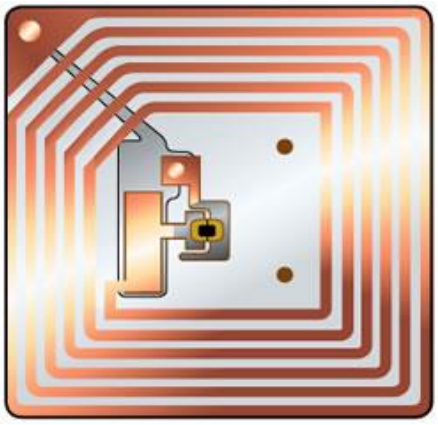
Tyyppi	Taajuus	Lukuetäisyys	Datanopeus	Sovellusalue
LF (Low Frequency)	30 - 300 kHz (125-134 kHz)	< 0.5 m	< 1 kbt/s	Eläintunnistus, kulunvalvonta
HF (High Freq.)	3 - 30 MHz (13.56Mhz)	max. 3 m	25 kbt/s	Kirjasto, kulunvalvonta
UHF (Ultra High Freq.)	300 MHz - 1 Ghz(EU 868 Mhz, USA 915 Mhz	max. 10 m	30 kbt/s	Logistiikka, ajoneuvotunnistus
Microwave	1 Ghz, 2.45 Ghz, 5.8 Ghz)	max. 10 m	max. 100 kbt/s	Ajoneuvotunnistus, rahtitunnistus



Kuvio 1. RFID-taajuusalueet (RFID frequency ranges. n.d.)

RFID-tunnisteet luokitellaan passiivisiin ja aktiivisiin. Tunnisteen tyypin mukaan skaalautuu sen käytettävyys ja hinta. Aktiivinen tunniste sisältää lähettimen ja virtalähteen, jonka antaman virran avulla tunniste toimii. Kuviossa 2 näkyy aktiivinen

RFID-tunniste. Virtalähde mahdollistaa passiivista tunnistetta laajemman lukukantaman ja halutun tiedon pidempiaikaisen säilyttämisen tunnisteen omassa muistissa. Passiivinen tunniste on kuitenkin pitkäikäisempi kuin aktiivinen tunniste siksi, että aktiivisen tunnisteen virtalähde voi joskus lakata toimimasta. (Violino. 2005.)



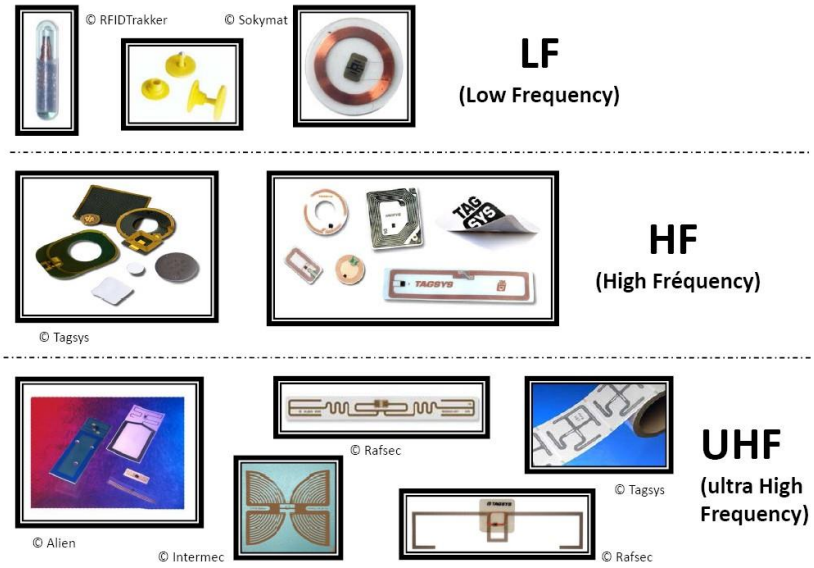
Kuvio 2. Aktiivinen RFID-tunniste (Violino. 2005.)

Aktiivinen tunniste voi toimia vastaanottajana tai lähettäjänä. Vastaanottajana tunniste aloittaa toimintansa silloin, kun lukijan signaali tavoittaa tunnisteen. Aktiivisen tunnisteen akkukesto on hyvä, koska tunniste on käytössä vain lukijan kentän ollessa tunnisteen kohdalla. Lähettävä tunniste lähettää tietoaan ympäristöön lukijalle määrätyssä intervallissa, esimerkiksi sekunnin välein tai kerran päivässä. (Violino. 2005.)

Passiivinen tunniste ei sisällä omaa lähetintä eikä virtalähdettä. Passiivinen tunniste saa virran lukijan lähettämästä radiosignaalista. Puolipassiivinen tunniste sisältää oman virtalähteen, mutta ei omaa lähetintään. Passiivitunnisteet, jotka käyttävät LF- tai HF-taajuuksia, käyttävät sähkömagneettista induktiota jännitteen ja virransiirrossa. Passiiviset UHF-alueen tunnisteet käyttävät sähkökentän heijastumista jännitteen ja virran siirrossa. Taulukossa 2 on eritelty tunnisteiden ominaisuuksia ja kuviossa 3 on nähtävillä tunnisteiden ulkonäkö käyttöalueesta riippuen. (Violino. 2005.)

Taulukko 2. RFID-tunnisteiden ominaisuuksia. (Violino. 2005.)

Tyyppi	Taajuus	Lukuetäisyys	Soveltuvuus	Hinta
Aktiivi	455 Mhz, 2.45 Ghz, 5.8 Ghz	20 m – 100 m	Ajoneuvotunnistus, rahtitunnistus	Kallis
Passiivi	125 - 134 kHz, 3 - 30 MHz, 2.45 Ghz	0.3 m – 5 m	Lähitunnistus, tuoteseuranta	Halpa



Kuvio 3. RFID-tunnisteiden käyttöalueet (RFID frequency ranges. n.d.)

Standardit

RFID-tekniikan standardit ovat tärkeitä, sillä useiden eri järjestelmien tulisi olla keskenään yhteensopivia. Esimerkiksi eri toimijoiden järjestelmien tulisi kyetä käyttämään samoja tunnisteita. Standardit auttavat siinä, että laitteet ja tunnisteet eivät mene hukkaan, jos jokin tietty toimittaja tai valmistaja lopettaa toimintansa. Silloin ei ole sitoutumista tiettyyn valmistajaan eikä laitteistoon. Standardit osittain määrittävät RFID-käytön aluekohtaisuuksia. ”US only” -laitteita ei tulis käyttää EU-alueella. (Violino. 2005.) (RFID-standardit. n.d.)

RFID-tekniikassa standardit usein liittyvät moneen eri asiaan. Tärkeimmät standardit määräävät tiedonvälitysprotokollan (air interface protocol) ja tunnisteen tietosisällön. LF-taajuusaluella on vain varattuja standardeja. Esimerkiksi eri kulunvalvontajärjestelmät on toteutettu suljettuina järjestelminä 125 kHz taajuudella. (Violino. 2005.) (RFID-standardit. n.d.)

HF-taajuusalueella 13.56 Mhz taajuudella taas on olemassa ennaltasovittuja standardeja. Esimerkiksi Standardi ISO14443 ei lupaa valmistajariippumatonta tunnisteiden ja lukijoiden yhteensopivuutta, eikä siten tuo kaikkia standardien etuja saataville. UHF-taajuusalueen yleisin standardi on ISO18000-6C eli Gen2-standardi. Se on EPC Global -järjestön kehittämä protokollastandardi. UHF-taajuuden tunnistusprosessi on saatu paremmaksi Gen2:n myötä. (RFID-standardit. n.d.) (Poole. n.d.)

2.3 NFC

NFC eli Near Field Communication tarkoittaa lähikenttäviestintää. NFC:tä voidaan sanoa RFID:n sivutuotteeksi. NFC on suunniteltu käytettäväksi erilaisiin etälukutoimintoihin, kuten etämaksamiseen e-lompakon avulla. NFC toimii todella pienillä etäisyyksillä ISM-taajuudella 13.56 Mhz. NFC:tä käyttävät laitteet voivat olla aktiivisia tai passiivisia. Esimerkki passiivisesta laitteessa on NFC-tunniste, joka sisältää ennaltakirjoitettua tietoa, jota ainoastaan muut laitteet voivat lukea. Aktiivilaitteet voivat taas sekä vastaanottaa että lähettää tietoa eteenpäin. Tälläinen NFC-aktiivilaite on esimerkiksi nykyajan matkapuhelin, jossa on NFC-valmius. NFC-lukija ja tunniste on esitetty liitteessä 11. (Near Field Communication n.d.) (Thrasher. 2013.)

Toiminta

Toiminnaltaan NFC käyttäytyy niin, että NFC-siru aktivoituu toisen sirun läheisyydessä ja tiedonvälitys tapahtuu kahden sirun välillä ilman yhteysparametrien vaihtoa. Esimerkki NFC:n käytöstä on kahden eri matkapuhelimen välinen tiedonvälitys vain puhelinten takakuoria toisiinsa koskettamalla. Tämän jälkeen tieto siirtyy puhelimesta toiseen NFC-tekniikan avulla. (Near Field Communication n.d.)

NFC, Bluetooth ja WiFi vaikuttavat toisiinsa nähden samankaltaisilta tekniikoilta. Kaikki kolme tekniikkaa mahdollistavat tiedonsiirron eri laitteiden välillä lyhyiden kantamien sisällä. Ero tekniikoiden välillä on siinä, että NFC käyttää elektromagneettisia kenttiä tiedonsiirtoon, kun taas Bluetooth ja WiFi käyttävät radioaaltoja tiedonvälitykseen. (Near Field Communication n.d.)

NFC-tiedonvälityksen tietoturvallisuus mahdollistetaan ohjelmallisesti. NFC:tä käyttävä sovellus pystyy määrittelemään kommunikaation tietoturvaparametrit (secure element), esimerkiksi pankkitilinumeron lähimaksua varten. Tälläiset parametrit tallennetaan yleensä sovelluksen muistiin, ja ne ovat siellä suojattuna. (NFC Frequently Asked Questions. n.d.)

Tyypit ja ominaisuudet

NFC-tunnisteet voidaan luokitella neljään eri tyyppiin niiden standardien ja ominaisuuksien mukaan. Tyyppi A, B, C (Felica) ja D. Eniten tyypit eroavat toisistaan muistikoon mukaan:

1. Type A: perustuu ISO/IEC 14443A -standardiin. Tunnisteen muisti voidaan uudelleenkirjoittaa ja lukea. Tunnisteen muistin kirjoitus pystytään estämään suojauksella. Muistikoko 96 bits - 2 kbits. Tiedonsiirtonopeus 106 kbits/s. Esimerkkitunniste: innovision Topaz

2. Type B: perustuu ISO/IEC 14443A -standardiin. Tunnisteen muisti voidaan uudelleenkirjoittaa ja lukea. Tunnisteen muistin kirjoitus pystytään estämään suojauksella. Muistikoko 48 bits - 2 kbits. Tiedonsiirtonopeus 106 kbits/s. Esimerkkitunnisteita: NXP Mifare Ultralight, NXP Mifare Ultralight.

3. Type C: perustuu japanilaiseen teollisuusstandardiin (Japanese Industrial Standard) X 6319-4. Tunniste on esikonfiguroitu joko uudelleenkirjoitettavaksi tai vain-luku-tunnisteeksi. Muistikoko 1 Mb. Tiedonsiirtonopeus 212 kbits/s. Esimerkkitunniste: Sony Felica.

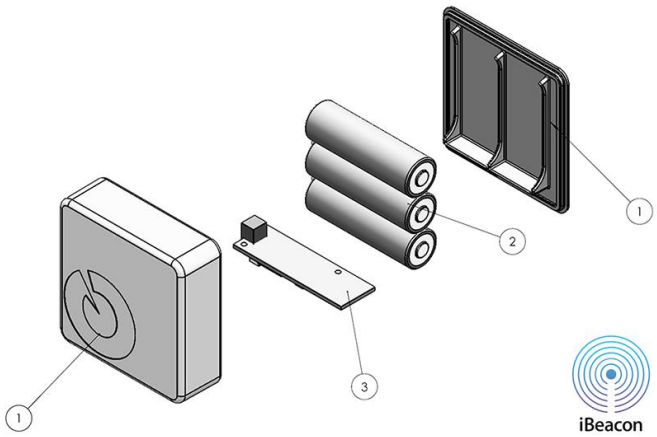
4. Type D: yhteensopiva ISO/IEC 14443 (A \& B) -standardien kanssa. Tunnisteen muisti voidaan uudelleenkirjoittaa ja lukea. Tunnisteen muistin kirjoitus pystytään estämään suojauksella. Muistikoko max. 32 kbits/s.

Tiedonsiirtonopeus 106 kbits/s. Esimerkkitunnisteita: NXP DESfire, NXP SmartMX

(NFC Tag Types. 2011)

2.4 iBeacon

iBeacon on Applen lanseeraama langaton tunnistustekniikka. Se on suhteellisen uusi tekniikka vuodelta 2012. Se käyttää hyväkseen toiminnoissaan Apple-laitteiden sisäistä Bluetooth-adapteria ja Applen iOS-käyttöjärjestelmän Location Services -palvelua. Kuviossa 4 on räjäytyskuva, josta voi nähdä erään iBeacon-laitteen osat: kotelon, akut ja piirilevyn. (Getting Started with iBeacon. 2014.)



Kuvio 4. Eräs iBeacon-laite (Roximity iBeacon. n.d.)

Toiminta

Kun Apple-laite havaitsee iBeaconin lähettämän signaalin, laite voi hälyttää käyttäjää. Esimerkiksi kauppakeskukseen saapuessa iBeacon tunnistaa saapuvan laitteen, ja puskee laitteen näytölle halutun viestin. Sijainnin määrityksessä iBeacon käyttää uusinta Bluetooth 4.0-tekniikkaa, jolla muut Apple-tuotteet havaitaan viestitystä varten. Bluetooth toimii 2.4 Ghz -taajuusalueella. (Getting Started with iBeacon. 2014.)

2.5 Bluetooth

Bluetooth on globalisoitunut langattoman tiedonsiirron standardi, joka mahdollistaa helpon, tietoturvallisen tiedonsiirron lyhyillä etäisyyksillä. Nykypäivänä se on yleisin tapa yhdistää erilaisia laitteita toisiinsa ja tietoympäristöön. (Bluetooth Basics n.d.)

Teknologian loi Ericsson-yritys vuonna 1994. Aluksi Bluetoothin tarkoitus oli olla korvaaja RS-232-datakaapeleille. Vuonna 1998 Ericsson, Intel, Nokia ja Toshiba muodostivat SIG:n (Bluetooth Special Interest Group), jonka tarkoituksena on jatkokehittää Bluetooth-tekniikkaa vielä nykypäivänäkin. Kukaan yksittäinen yritys ei omista Bluetooth-tekniikkaa. (Bluetooth Basics n.d.)

Bluetooth-termin alkuperä juontaa juurensa 1100-luvulle Tanskalaiskuninkaan Harald Blåtandin aikaan. Blåtand-sanan englanninkielinen käännös on Bluetooth. Harald auttoi yhdistämään sotaisat järjestöt sen aikaisessa maailmassa, nykyisen Norjan, Ruotsin ja Tanskan alueilla. Historiaa matkimalla Bluetooth-teknologia luotiin avoimeksi standardiksi yhdistämään toisiinsa eri laitteita, ihmisiä ja yrityksiä. (Bluetooth Basics n.d.)

Toiminta

Bluetooth-tekniikka mahdollistaa tiedonvälityksen lyhyillä kantamilla radioaaltojen avulla. Bluetooth toimii ISM-taajuuksilla välillä 2.4 - 2.485 Ghz. Se käyttää hajautettua spektriä, taajuushyppelyä ja full-duplexia signalointiin. Taajuushyppelynopeus on 1600 hyppyä/s. Hajautettu spektri ja täys-duplex mahdollistavat viestin molemminpuolisen lähettämisen parametreilla, jotka vain yhteyden vastapuolella on tiedossa. Viesti voidaan lähettää käyttäen suurta taajuusaluetta, jolloin tiedonsiirtokin on nopeaa. Bluetooth on hieno yhdistelmä hardwarea ja softwarea. Bluetooth-järjestelmä sisältää lähetinosan, jonka radion kautta tieto liikkuu ympäristöön. Sovellusosa puolestaan neuvottelee ja kommunikoi toisten Bluetooth-sovellusten kanssa. (Bluetooth Basics n.d.)

Bluetoothia käytetään miljoonissa eri tuotteissa. Autoissa, kännyköissä, kelloissa, lääketieteellisissä laitteissa, viihde-elektroniikassa ja tietokoneissa hyödynnetään Bluetooth-tekniikkaa nykypäivänä. (Bluetooth Basics n.d.)

Yhteyden muodostus

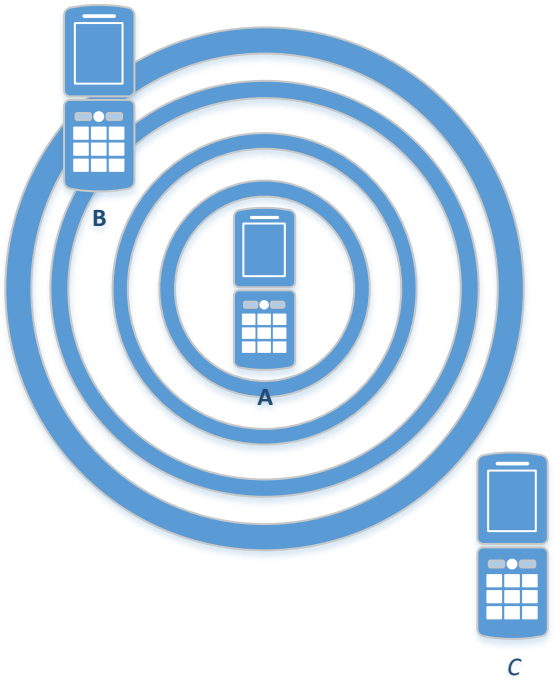
Bluetooth-yhteyden muodostuksen vaiheet ovat seuraavat:

1. Laite A lähettää ympärilleen kaikille mahdollisille Bluetooth-laitteille *Inquiry*-viestiä, kuten kuviossa 5.

2. A-laitteen kuuluvuusalueen sisällä oleva laite B vastaa *Inquiry Response* -viestillä, joka sisältää laitteen B Bluetooth-osoitteen.

3. Laite A listaa laitteet, joilta vastaus on tullut. Tämän jälkeen A lähettää yhteydenmuodostuspyynnön laitteelle B. Laite C ei ole kantaman sisällä, joten sille ei lähetetä yhteydenmuodostuspyyntöä.

(Overview of Bluetooth Pairing. 2008.)



Kuvio 5. Bluetooth-laitteiden hakualue

2.6 Tietokantarajapinta

Tietokanta

Tietokanta on kokoelma tietoa eli dataa. Se on yhtenäinen kokoelma, jossa tiedolla on merkitys ja liityntä toiseen tietoon. Tietokannan tarkoitus on säilyttää käyttäjän tietoa. Tietokantaan voidaan tallentaa suuriakin määriä tietoa, ja hakea sitä sieltä käyttäjän hakukriteerien mukaisesti. Tietokanta luodaan yleisesti johonkin tiettyyn tarkoitukseen, esimerkiksi asiakastietokannaksi, henkilöstötietokannaksi tai tuotetietokannaksi. (Jaakkola & Sarja. 2006.)

Tietokannassa tiedot kerätään tauluihin ja tauluissa riveihin ja sarakkeisiin. Kuviossa 6 on esitetty taulun rakenne. Sarakkeet ovat taulun kentät ja rivit ovat tietueet. Tietue muodostuu yhdestä tai useammasta rivin kentästä. Riviä voidaan siis kutsua tietueeksi. Taulun tiedon tyyppi määräytyy kenttien tietotyypin mukaan. Tietotyyppi voi olla esimerkiksi numeerinen arvo, tekstiarvo, päivämääräarvo tai kellonaika-arvo. Tietokannan tieto muodostuu taulun tietueiden arvoista. (Kilpeläinen. 2007.) (Relaatiotietokantojen peruskäsitteet. 2004.)

	Taulu 1			
	Sarake 1	Sarake 2	Sarake 3	
Rivi 1	Kenttä	Kenttä	Kenttä	Tietue 1
Rivi 2	Kenttä	Kenttä	Kenttä	Tietue 2
Rivi 3	Kenttä	Kenttä	Kenttä	Tietue 3
Rivi 4	Kenttä	Kenttä	Kenttä	Tietue 4
Rivi 5	Kenttä	Kenttä	Kenttä	Tietue 5

Kuvio 6. Tietokannan taulun osat

Taulujen kentille voidaan antaa attribuutteja, jotka vaikuttavat kenttien sallittuihin arvoihin ja kentän toimintoihin. Näitä ominaisuuksia ovat:

1. Tietueen sarake on perusavain (primary key)
2. Tietueen sarake on viiteavain (secondary key)
3. Tietue ei voi olla tyhjä (not null)
4. Tietueella on oletusarvo (default)

5. Tietueen arvo on yksilöllinen (unique)

Tietokantoja on montaa eri tyyppiä. Tietokannan rakenne ja toiminta määräytyvät yleisesti tietokannan tyypin mukaan. Tietokantojen tyypit ovat:

1. Relaatiotietokanta
2. Verkkotietokanta
3. Oliotietokanta
4. Oliorelaatiotietokanta
5. Multimediatietokanta
6. Hierarkkinen tietokanta

(Häkkinen. 2013.)

Esimerkiksi relaatiotietokannassa useat eri taulut ovat yhteydessä toisiinsa. Taulujen välisten yhteyksien (relaatioiden) avulla tiedon tallennus ja päivitys on nopeaa, koska tiedot pyritään lisäämään tauluihin niin, että yksi tieto lisätään vain yhteen paikkaan. Relaatiotietokantaan talletetaan tieto myös siitä, miten eri taulukot liittyvät toisiinsa. Yleensä tietokantaa hallitaan jollakin tietokannan hallintasovelluksella, kuten Access, Paradox, Oracle, PostgreSQL tai MySQL. (Häkkinen. 2013.) (Kilpeläinen. 2007.) (Sarja. 2006.)

Relaatiotietokantojen suunnittelussa huolellisuus on hyve. Hyvin suunniteltu tietokanta on nopea ja selkeäkäyttöinen, sekä palvelee sen käyttötarkoitustaan oikealla tavalla. Myös turhalta työltä välttyy kun kerran suunnitellaan tietokanta hyvin. Turhat tietojen syötöt, ristiriitaisuudet ja virheilmoitukset voidaan välttää. (Sarja. 2006.)

Avaimet

Yleensä tietokannan taulun jokin tietotyyppi on määritelty avaimeksi. Avaintyyppejä ovat perusavain ja viiteavain. Perusavain on pakollinen jokaisessa taulussa, ja

viiteavainta käytetään silloin, jos tuodaan tieto jostain muusta taulusta määrättyyn, viiteavaimelle varattuun kenttään. (Kilpeläinen. 2007.)

Perusavain on sarake, jonka perusteella rivit tunnistetaan ja erotetaan toisistaan. Perusavaimeen liittyy muutama sääntö:

1. perusavain on yksikäsitteinen ja muodostuu yhdestä tai useammasta sarakkeesta
2. jokaisella taulun rivillä perusavaimella on oltava eri arvo
3. perusavaimen arvo ei saa puuttua yhdeltäkään riviltä
4. perusavaimen eheys (entity integrity)

(Kilpeläinen. 2007.)

Viiteavain (foreign key) on sarake, jota käytetään luomaan relaatioita eli suhteita taulujen välille. Viiteavainsarake luodaan tauluun, jossa tyypillisesti viitataan toisessa taulussa olevaan riviin tämän toisen taulun jonkin rivin perusavainta vastaavalla arvolla. Näin saadaan linkitettyä eri tauluissa olevat tiedot yhteen. Viiteavaimeen liittyviä sääntöjä ovat:

1. viiteavaimen arvoalue on sama kuin viitattavan taulun perusavaimen arvoalue
2. sarakkeet ovat vastaavat kuin viitattavassa taulussa
3. viite-eheys

(Kilpeläinen. 2007.)

Tietokannan eheys

Tietokannan eheys tarkoittaa tietokannan tiedon keskinäistä yhteensopivuutta ja käyttäjän asettamien ehtojen oikeellisuutta. Eheys voidaan määritellä kolmen vaatimuksen avulla: avaineheys, viite-eheys ja attribuuttieheys. Jos tietokannan taulussa on pääavain ja sen arvo on yksilöllinen, tietokannan avaindata on eheä. Pääavaimen ja viiteavaimen välinen oikea toimiva viittaus on viite-eheys. Viiteavaimen vastine on tällöin toisen taulun pääavain. Attribuuttieheys on kunnossa

silloin, kun tietueen määrätty arvo vastaa kentän asetettua parametriä. (Häkkinen. 2013.)

Tietokantahallintajärjestelmä

Tietokannan hallintajärjestelmä eli DBMS (Database Management System) on ohjelmisto tietokannan ytimen (database engine) käyttämiseen, tietokantojen luomiseen ja hallinnointiin. DBMS voi yleensä sisältää käyttäjälle valmiiksi optimoituja kyselyjä, indeksejä ja apuohjelmia helpottamaan käyttökokemusta. Usein graafinen käyttöliittymä myös auttaa tietokannan rakenteen hahmottamisessa ja tekee tietokantatyöskentelystä helpompaa. Tietokannan käyttäjän ja tietokannan osien suhteet on kuvattu kuviossa 7. (Häkkinen. 2013.)

Kuvio 7. Käyttäjän ja tietokannan osien suhteet

Tietokannan hyödyt

Tietokantojen käyttö antaa useita etuja tietoteknisiin ratkaisuihin. Tietokanta on huomattavasti nopeatoimisempi kuin esimerkiksi Windows Excel -taulukkolaskentasovellus tai fyysinen paperiarkisto teksteineen. Tietokannan tiedot ovat nopeasti saatavilla usealle käyttäjälle, myös verkon ylitse. Tällöin tietokanta voi olla hyvinkin siirrettävä, jos se sijaitsee esimerkiksi verkkopalvelimella. (Häkkinen. 2013.)

Tietokannan tiedon esitys on todella joustavaa, ja se voidaan luoda mukautetuksi näkymäksi käyttäjän tarpeiden mukaan. Tietokanta on yleensä pitkäikäinen ja kustannustehokas. Hyvin suunniteltu tietokanta on halpakustanteinen suuren tietomäärän varastointiin verrattuna. (Häkkinen. 2013.)

Tietokanta on mahdollista tehdä myös tietoturvalliseksi. Tiedon tallentaminen ja hakeminen voidaan salata, ja tietokannan käyttöön voidaan määrittää salasanat. Käyttäjäkohtaisia rajoitteita on myös mahdollista asettaa. (Häkkinen. 2013.)

2.7 SQL

SQL eli Structured Query Language on rakenteinen kyselykieli. SQL on myös ANSI-standardi. Tämä tarkoittaa sitä, että käyttäjällä on aina käytössä tietty komentojen perusjoukko, joka toimii kaikissa eri tietokantaratkaisuissa, kuten MySQL- tai Microsoft Access -tietokannoissa. (SQL-alkeet. n.d.)

SQL:ää käytetään, kun käsitellään ja haetaan tietoja relaatiotietokannasta. Kaikki keskeiset relaatiotietokantoja hyväkseen käyttävät ratkaisut tukevat tai hyödyntävät SQL:ää jollain tapaa. SQL:ää voidaan hyödyntää niin yhden käyttäjän tietokantaa käyttämässä Windows-sovelluksessa kuin useiden tuhansien käyttäjien SQL Server -ohjelmassa. SQL-kielen avulla käyttäjä voi:

1. muokata tietokannan rakennetta
2. muuttaa järjestelmän turva-asetuksia
3. lisätä käyttäjän oikeuksia tietokantaan
4. suorittaa kyselyjä tietokantaan
5. päivittää tietokantaa

(SQL-alkeet. n.d.)

Käskyjä tietokannan käyttöön ovat mm. *SELECT, UPDATE, INSERT* ja *DELETE*. Tietokannan tiedon käsittelyyn käskyjä ovat *CREATE TABLE* ja *CREATE VIEW*, joilla luodaan uusia tietokantaobjekteja kuten taulu ja näkymä. *ALTER*-käskyä käytetään jonkin tietokantaobjektin muokkaamiseen jälkeenpäin. Esimerkiksi tauluun voidaan lisätä uusi sarake myöhemmin. (SQL-alkeet. n.d.)

SQL-lauseet

Esimerkiksi lukijaksriptin SQL-lause tietokantaan on seuraavanlainen

INSERT IGNORE INTO rfidtaulu (rfidkolumni) ' + 'VALUES("' + tag + '")';

Lauseen *”INSERT IGNORE INTO”* -osa määrittää halutun tiedon lisäyksen tauluun siten, että samanlaisia arvoja ei huomioida. Tauluun ”rfidtaulu” ja sen kolumniin ”rfidkolumni” lisätään arvo ”tag”, joka on lukijan lukema tunniste.

Tietokannan sisällön näyttäminen tapahtuu *SELECT * FROM rfidtaulu* -lauseella. *”SELECT * FROM”* määrittää mitkä kentät taulusta halutaan näyttää. *-merkki määrittää näytettäväksi kaikki kentät. Tarkemmat parametrit sisällytetään hakuun siinä järjestyksessä, missä ne esiintyvät taulussa.

SELECT rfidkolumni, rfidtime, rfidid FROM rfidtaulu

Tietokannan tiedot voidaan tyhjentää *DELETE* -lauseella. Sille voidaan myös määrittää lisäparametreja tarkempaa tyhjennystä varten. Toinen mahdollinen tyhjennyslause on *TRUNCATE*. Se tyhjentää kaikki halutun taulun tiedot, ja jättää jäljelle tyhjän taulun.

TRUNCATE TABLE rfidtaulu

2.8 MySQL

MySQL on avoimen lähdekoodin tietokantahallintasovellus (DBMS). MySQL mahdollistaa tietokannan hallinnoinnin. Tietokantaa voidaan hallita yleensä lokaalilta- tai verkkopalvelimelta tai erilliseltä hallintakoneelta. Hallintasovelluksessa on usein graafinen käyttöliittymä, jonka avulla tietokantaa hallitaan. MySQL tukee myös CLI-muotoista tietokannan hallintaa ja sen syntakseja. (What is MySQL? n.d.)

2.9 PHP

PHP eli Hypertext Preprocessor on avoimen lähdekoodin ohjelmointikieli. PHP on laajasti käytetty, lähinnä HTML-koodin yhteydessä. PHP:llä saadaan Internet-sivustoille dynaamista sisältöä, esimerkiksi painikkeita, bannereita, taulukoita ja

ikkunoita. PHP muistuttaa C-, Java- ja Perl-kieliä syntakseiltaan, joten se lukeutuu tehokkaisiin ja suhteellisen helposti opittaviin ohjelmointikieliin. (Mikä on PHP? n.d.)

Historia

PHP-kielen kehittäminen alkoi vuonna 1994. Tanskalainen Rasmus Lerdorf tutki ja kirjoitti C- ja Perl-kielisiä skriptejä kotisivujaan tutkiessaan, ja näiden kokeilujen myötä syntyi alkuperäinen PHP eli Personal Home Page Tools. Vuonna 1995 Lendorf jalosti keksintöään lisää, ja tuloksena syntyi PHP/FI eli Form Interpreter. FI:llä pystyi jäsentämään tietoa verkkosivun lomakkeista, siis saattoi käyttää tietokantoja verkkosivujen avulla. (Mikä on PHP? n.d.)

PHP/FI 2.0 julkaistiin 1997. 2.0-versiolla oli useita tuhansia käyttäjiä ympäri maailmaa. Silloin ohjelma oli asennettu noin 50 000:een domainiin (1 % kaikista Internetin domaineista) ja useat ihmiset osallistuivat sen kehittämiseen. Projektin päävastuu oli yhä Rasmus Lerdorfilla. (Mikä on PHP? n.d.)

Lerdorfin kollegat Andi Gutmans ja Zeev Suraski totesivat PHP/FI 2.0:n riittämättömäksi monimutkaisempien asioiden toteuttamiseen. He kirjoittivat lähes koko lähdekoodin alusta asti uudelleen ja julkaisivat sen kesäkuussa 1998 nimellä PHP 3. Lyhenteelle tuli uusi nimitys, PHP: Hypertext Preprocessor. (Mikä on PHP? n.d.)

Talvella 1998 Gutmans ja Suraski aloittivat PHP:n ytimen uudelleenkirjoittamisen. Heidän tavoitteenaan oli luoda ydin, joka tukisi kolmansien osapuolten ohjelmointirajapintoja. Uusi ydin julkaistiin vuonna 1999 nimellä Zend Engine (yhdistelmä nimistä Zeev ja Andi), ja maaliskuussa 2000 julkaistu PHP 4 käyttää sitä edelleen ytimenään. PHP 4:n virallinen tuki päättyi 8. elokuuta 2008. (Mikä on PHP? n.d.)

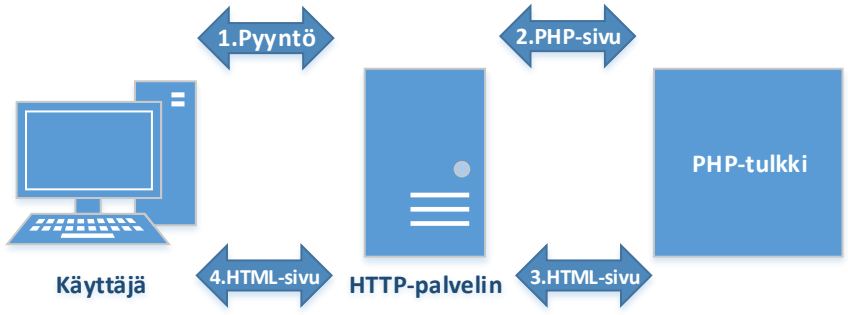
PHP 5 julkaistiin heinäkuussa 2004. Sen ytimenä on Zend Engine II, joka muun muassa tukee olio-ohjelmointia ja sisältää sisäänrakennetun tietokantamoottorin, SQLiten. (Mikä on PHP? n.d.)

Toiminta

PHP-ohjelmointiin tarvitaan tekstieditori, web-palvelin ja web-selain. ASCII-tekstieditorilla pystyy luomaan PHP-koodia. Tälläinen tekstieditori on esimerkiksi Windows-käyttöjärjestelmän Muistio (Notepad) -sovellus. PHP-ohjelmointiin on myös saatavilla varta vasten tehtyjä ohjelmistoja. (Mikä on PHP? n.d.)

Jotta PHP-ohjelmoitua ohjelmaa voidaan suorittaa, tarvitaan siihen Web-palvelinohjelmisto sekä verkkoselain. Verkkoselain toimii PHP-tulkkina. Palvelinohjelmistoa tarvitaan verkkoyhteyden luomiseen ja hallinnointiin sekä tiedostojen hallintaan. Palvelinohjelmiston PHP-tulkki taas lukee ja ymmärtää kirjoitettua PHP-koodia ja osaa näyttää sen visuaalisesti käyttäjälle. PHP:llä voi dynaamisten web-sivustojen lisäksi tehdä myös komentorivisovelluksia ja jopa graafisia käyttöliittymiä, mutta sen toiminnallisuus on tarkoitettu web-sovelluskehitykseen. (Mikä on PHP? n.d.)

Normaalitilanteessa käyttäjän pyytäessä palvelimelta staattista HTML-sivua palvelin yksinkertaisesti palauttaa käyttäjälle pyydetyn sivun. PHP-sivuilla kaiken toiminnallisuuden toteuttava koodi kirjoitetaan HTML-koodin sekaan. Sivut nimetään tavallisesti .php-päätteisiksi, mutta ne voidaan nimetä myös esim. .php3 tai .php4. Tällöin ilmaistaan samalla PHP:n versio. HTTP-palvelimen asetuksissa määritellään, minkä päätteen omaavat tiedostot lähetetään PHP-tulkille käsiteltäväksi. PHP-sivua kutsuttaessa palvelin toimii kuvion 8 mukaisesti. (Mikä on PHP? n.d.)



Kuvio 8. PHP-prosessi

Http-palvelin tunnistaa palvelua pyydettäessä tiedoston päätteen perusteella, että kyseessä on tosiaan PHP-sivu. Siinä tapauksessa palvelin välittää ensimmäisenä PHP-tulkille pyynnön esiprosessoida pyydetty sivu. PHP-tulkki kääntää ja suorittaa

tiedoston sisältämän PHP-koodin. Lopputuloksena PHP-tulkki palauttaa http-palvelimelle pelkkää HTML-koodia. Lopuksi http-palvelin palauttaa asiakkaalle HTML-koodin. PHP-koodi on siis olemassa ainoastaan palvelimella. Käyttäjä ei näe sitä missään vaiheessa. (Mikä on PHP? n.d.)

2.10 Node.js

Node.js NPMNode.js on avoimeen lähdekoodiin perustuva ajoympäristö. Node.js on pohjimmiltaan alusta, jolla on tarkoitus rakentaa helposti ja nopeasti skaalautuvia verkkosovelluksia. Se on tarkoitettu käytettäväksi palvelimissa ja verkkosovelluksissa. Node.js-skripti kirjoitetaan JavaScriptillä. Node.js on järjestelmäriippumaton ja sitä voidaan suorittaa esimerkiksi OS X-, Windows- ja Linux-ympäristöissä. Node.js käyttää Google Chromen V8 Javascript Engineä JavaScriptin parserointiin. V8 on nopea verrattuna esimerkiksi Python- tai Perl-parsereihin. (NodeJS.org. n.d.)

Esimerkiksi kuvion 9 mukainen skripti luo palvelimen osoitteeseen 127.0.0.0:1337, jossa käyttäjä voi nähdä ”Hello World” -tekstin.

```
var http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, "127.0.0.1");

console.log('Server running at http://127.0.0.1:1337/');
```

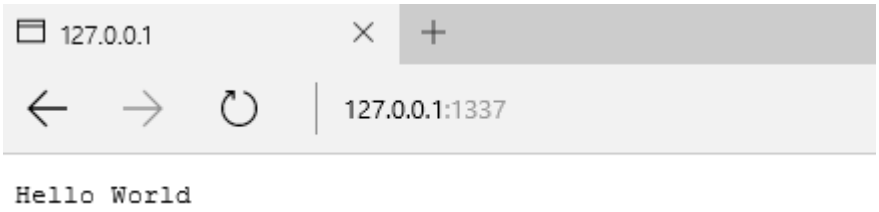
Kuvio 9. hello.js Node.js-skripti (NodeJS.org. n.d.)

Hello.js ajetaan cmd.exen tai node.exen kautta kuvion 10 mukaisella syntaksilla. Skripti ei sulkeudu ajettuaan koodin; se jää taustalle odottamaan, jos sillä ei ole mitään tehtävää.

```
C:\>node hello.js
Server running at http://127.0.0.1:1337/
```

Kuvio 10. "Hello World" -tuloste

Käyttäjä näkee “Hello World” -tervehdyksen verkkoselaimessaan kuvion 11 tapaan siirtymällä osoitteeseen 127.0.0.1:1337, joka on ohjelman luoman palvelimen osoite.



Kuvio 11. "Hello World" -selainnäkymä

NPM

NPM eli Node Package Manager on Internetissä oleva repositorio. NPM tarjoaa pakettienhallintaa Node.js:lle ja JavaScriptille. NPM:n avulla voidaan julkaista avointa lähdekoodia verkkoon jaettavaksi. NPM antaa JavaScript-kehittäjille mahdollisuuden jakaa ja käyttää hankittua koodia helposti. NPM mahdollistaa tarvittavien lisäosien ja moduulien asentamisen kätevästi yhdellä komennolla komentokehotteen (CLI) kautta. NPM on maailman suurin avoimen lähdekoodin kirjastoja sisältävä repositorio (NodeJS.org. n.d.) (Reed. 2011.)

2.11 JavaScript

JavaScript on oliopohjainen ohjelmointikieli, jolla tehdään verkkosivustoista interaktiivisia. JavaScriptin on kehittänyt Netscape Communications Corporation vuonna 1995. JavaScript-tiedostopääte on ”.js”. JavaScript toimii käyttäjän selaimessa, eikä tällöin tarvitse palvelinpuolen resursseja toimiakseen. Esimerkki JavaScriptistä verkkosivulla on monivalintakysely ja siihen liittyvä vastauspainike. Myös pop-up -ikkunat voidaan toteuttaa JavaScriptin avulla. (Chapman. n.d.) (JavaScript-perusopas. 2007.)

JavaScript ei ole sama kuin Java. Molemmissa kielissä on samankaltaisuuksia, mutta ei muuta yhteistä. JavaScriptin käyttö tapahtuu normaalisti selaimessa. Java tarvitsee oman tiedoston tai ohjelman eri toimintoihin. (Chapman. n.d.)

Useimmista verkkoselaimista saa JavaScript-tuen pois päältä. Jotkut käyttäjät kytkevät JavaScript-tuen pois päältä tietoturvariskien, mahdolliseen JavaScript-pohjaisten haittaohjelmien pelosta. Monet eivät myöskään pidä pop-up -ikkunoista ja eri JavaScript-pohjaisista animaatioista verkkosivustoilla. (JavaScript-perusopas. 2007.)

Toiminta

JavaScript-ohjelmoinnin aloittamiseen tarvitaan JavaScriptiä tukeva verkkoselain ja tekstieditori, kuten notepad.exe. Automaattinen rivitys voi aiheuttaa harmia koodirivien katkeamisen muodossa. JavaScript-koodia voidaan kirjoittaa suoraan HTML-dokumentin sisään. Määrityksien <script> ja </script> väliin tulee itse JavaScript-koodi. (JavaScript-perusopas. 2007.)

Esimerkki ”Hello World” -Javascript-koodi, joka selaimessa ajettuna näyttää tekstin ”Hello World”:

```
!DOCTYPE HTML>
<html>
<body>
 <p>Header...</p>
 <script>
  alert('Hello, World!')
 </script>
 <p>...Footer</p>
</body>
</html>
```

(JavaScript-perusopas. 2007.)

2.12 Python

Python on ohjelmointikieli. Se on monipuolinen ominaisuuksiltaan ja siitä on pyritty tekemään yksinkertainen ja havainnollinen rakenteeltaan. Se muistuttaa olio-ohjelmointikieltä, jossa ohjelmoija voi määritellä datarakenteille operaatioita. Näitä operaatioita kutakin datarakennetta vastaava olio sitten “tekee”. (General Python FAQ. n.d)

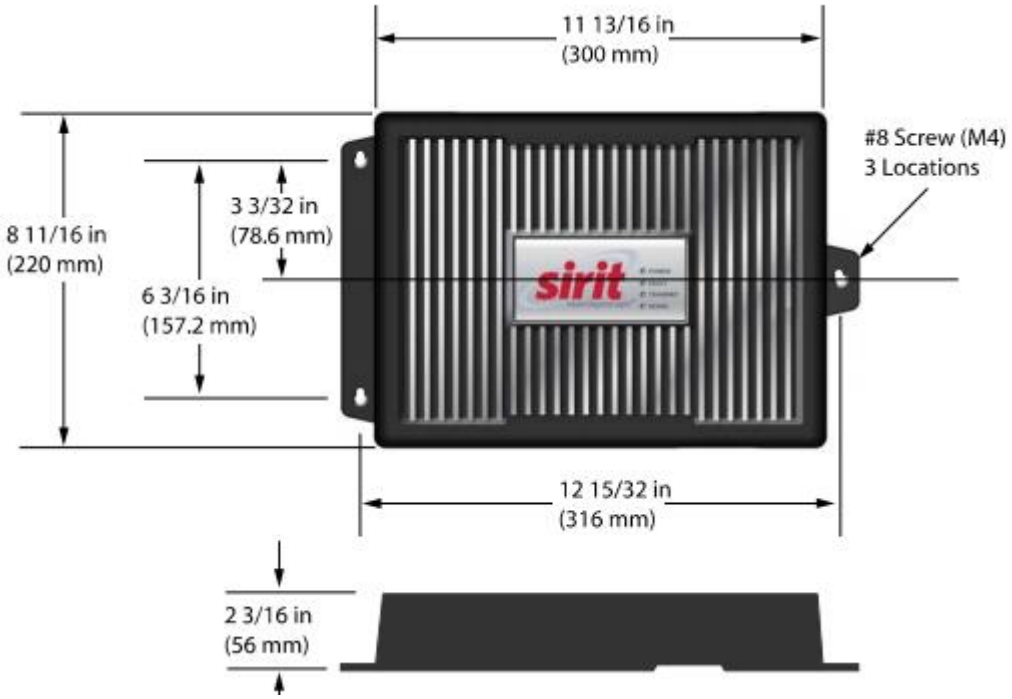
Pythonissa on monia piirteitä Perl-ohjelmointikielestä. Python on tulkittava (interpreted)-ohjelmointikieli; Python-kielisen ohjelman suorittamiseen tarvitaan Python-tulkkiohjelma, joka lukee koodia ja suorittaa koodissa määrätyt käskyt. Pythonia voidaan käyttää useilla eri käyttöjärjestelmillä, kuten Mac ja PC Windows. (General Python FAQ. n.d)

2.13 Xampp

XAMPP on PHP-kehitysympäristö. Se on täysin ilmainen sovellus, joka perustuu avoimeen lähdekoodiin. Se on jakelupaketti, joka sisältää Apache-webpalvelimen ja MySQL:n sekä PHP- ja Perl -kirjastot. Se on suosittu monimuotoisuutensa ja helppokäyttöisyytensä vuoksi. XAMPP tarjoaa aloitustyökalut omien verkkosivujen, tietokantojen ja niiden elementtien rakentamiseen. (What is XAMPP? n.d.)

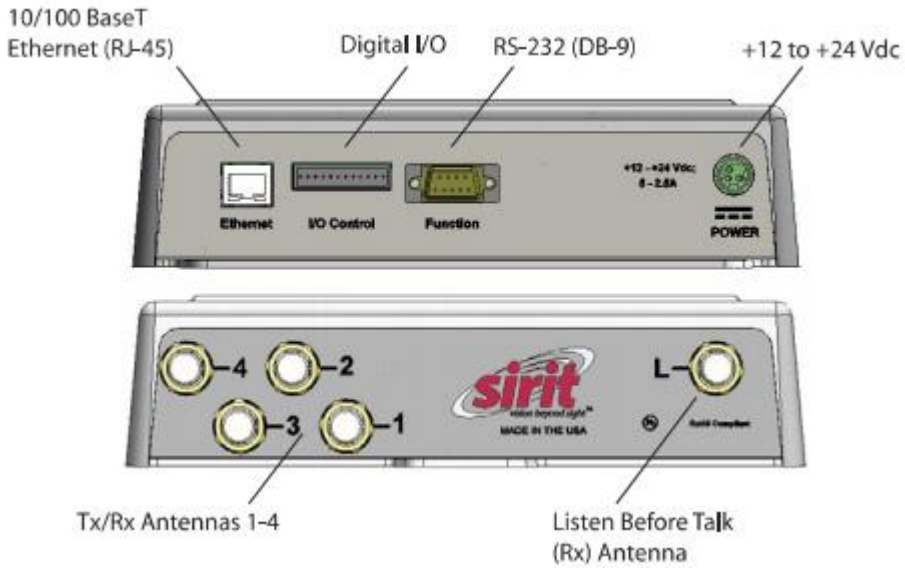
2.14 SIRIT INfinity 510

Sirit Infinity 510 RFID-lukija on Sirit-yrityksen valmistama RFID-lukija. Sirit on kanadalainen RFID-laitevalmistaja. Sirit lopetti Infinity 510-lukijoiden valmistuksen kesällä 2013. Suomeen Siritin lukijoita on tuonut toimialaratkaisuja tarjoava Finn-ID. Sirit 510 toimii UHF-alueen taajuuksilla välillä 860 - 960 Mhz. Lukija sopii käytettäväksi useiden eri maiden radiotaajuussäädösten kanssa. Hyvinä esimerkkeinä tästä toimivat Pohjois-Amerikka, Eurooppa ja Aasia. 510-lukija tukee Gen2- eli ISO 18000-6C -standardia ja sen mukaisten protokollien ominaisuuksia. Lukijan voi ajatella multiprotokollalukijaksi. Lukija tukee myös laiteohjelmiston päivitystä, jonka avulla laitteen ominaisuuksia ja protokollatukea voidaan lisätä ohjelmistopäivityksillä. Valitettavasti ohjelmistopäivityksiä ei enää ole saatavilla, koska Sirit lopetti tuen päivityksille laitteen valmistuksen loputtua. Kuviossa 12 on esitetty lukijan fyysiset mitat. (RFID-lukijat. n.d) (Sirit Infinity 510 User’s Guide. 2009.)



Kuvio 12. Sirit INfinity 510 RFID-lukija (Sirit Infinity 510 User's Guide. 2009.)

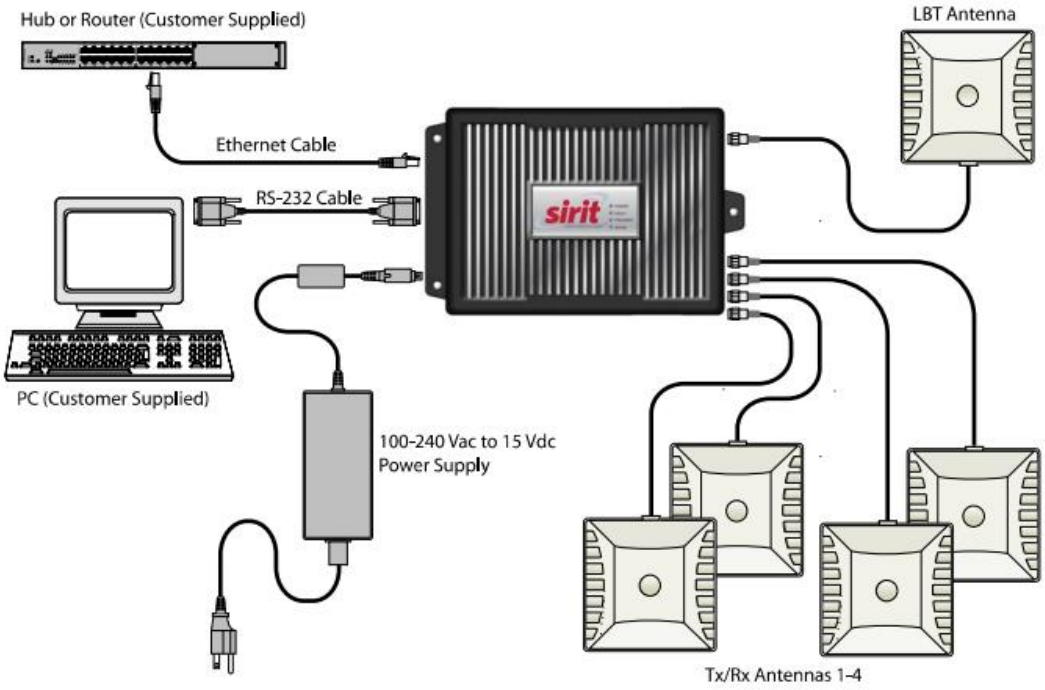
510-lukijaan on mahdollista kytkeä maksimissaan neljä Tx/Rx -antennia, joilla voidaan lähettää ja vastaanottaa dataa. Lukijassa on myös liitäntä yhdelle vain vastaanottavalle LBT-antennille. Antennit voidaan kiinnittää suoraan antenniportteihin tai käyttää lukijan ja antennin välistä antennikaapelia. Kaapelin maksimipituuden olisi hyvä olla 10 metriä. Lukijan tiedonsiirto mahdollistetaan Ethernet RJ-45 -liittimen, RS-232 DB-9 -sarjaportin tai Digital I/O -liittimen kautta. Lukijan hallinta, asetusten säätö sekä yhteys toisen laitteen kanssa tapahtuu näiden porttien kautta. Lukijassa on myös virtapistoke, ja lukijaa pystytään käyttämään +12 - +24 voltin käyttöjännitteellä. Lukijaa on mahdollista käyttää -20° C - 55° C asteen lämpötiloissa. Lukija painaa 3 kg ja sen kuori on valmistettu alumiinista. Lukijan liitännöistä voi saada paremman käsityksen katsomalla kuviota 13. (Sirit Infinity 510 User's Guide. 2009.)



Kuvio 13. Sirit INfinity 510 RFID-lukijan liitäntöjä (Sirit Infinity 510 User's Guide. 2009.)

Lukijan voi kiinnittää kotelon sivuilla olevien kiinnitysreikien avulla eri pintoihin, kuten betoniin, puuhun, metalliin ja lastulevyyn. Lukijan ohjekirjassa suositellaan käytettäväksi tiettyjä kiinnitysruuveja niille sopivien pintojen kanssa. (Sirit Infinity 510 User's Guide. 2009.)

Antennien kiinnitys lukijan ympärille on mahdollista mitoittaa käyttäjän halujen ja tarpeiden mukaan. Yhden ja kahden antennin RFID- järjestelmät sopivat liukuhihna-tyyliseen tunnistelukuun, jossa luettavia kohteita on vain tietyllä suunnalla lukijaan nähden. Kolmen ja neljän antennin järjestelmä on sopivampi portaali- eli porttikäyntimalliseen tunnistamiseen esimerkiksi ovien ja uloskäyntien yhteydessä. Lukijan yleisen liitäntöjen topologian voi nähdä kuviosta 14. (Sirit Infinity 510 User's Guide. 2009.)



Kuvio 14. RFID-järjestelmän topologia (Sirit Infinity 510 User's Guide. 2009.)

Antennien fyysinen sijoituspaikka sekä ohjelmistollinen säätö vaikuttavat lukukentän kokoon ja voimakkuuteen. Käyttämättömät antenniportit on usein syytä sulkea ohjelmallisesti, jolloin estetään lukijalaitteen vahingoittuminen. Portteihin ei saisi koskea käsin, koska ihmisen staattinen varaus voi vahingoittaa niiden kautta lukijaa tai porttia itseään. (Sirit Infinity 510 User's Guide. 2009.)

Lukijan koteloon on sijoitettu ilmaisinledejä kuvion 15 mukaisesti helpottamaan käyttäjän havainnointia lukijan toiminnasta. Ledejä on neljä erilaista ja ne ilmaisevat eri asioita:

1. ”Sense” ilmaisee lukijan havainneen tunnisteen.
2. ”Transmit” ilmaisee lukijan lähettimen toimivan oikein
3. ”Fault” ilmaisee lukijassa tapahtuneen virheen
4. ”Power” ilmaisee lukijan olevan virtapistokkeessa kiinni

(Sirit Infinity 510 User's Guide. 2009.)

Kuvio 15. Sirit INfinity 510 RFID-lukijan ilmaisinledit (Sirit Infinity 510 User's Guide. 2009.)

Sirit Infinity 510 -lukijassa on mahdollista ajaa käyttäjän luomaa ohjelmistokoodia tai skriptiä. Lukija tukee C++-, Java- ja Python-kieliä, joilla ohjelmia voidaan kirjoittaa. Tällöin lukija ei tarvitse erillistä tietokonetta ja hallintasovellusta toimiakseen, ja lukija voikin myös toimia kirjoitetun koodin avulla ilman käyttäjän syötettä itsenäisesti. Koodilla pystytään määrittämään esimerkiksi tunnisteluku ja kirjoitus, tiedonsiirto tietokantaan, automaattinen virrankatkaisu ja odotusajastimet lukutapahtumiin. (Sirit Infinity 510 User's Guide. 2009.)

Sirit Infinity 510 -lukijalle on mahdollista syöttää komentoja TCP-porttiin 50007. Se on kaksisuuntainen komentokanava. Käyttäjä syöttää tavallista ASCII-muotoista tekstiä lukijan hallitsemiseen hallintaohjelmiston tai komentorivin kautta. Verkkopohjaisen yhteyden kautta lukijaan käyttäjä voi syöttää myös tismalleen samanlaisia tekstikomentoja lukijan hallitsemiseen. (Sirit Infinity 510 Protocol Reference Guide. 2009.)

Portti 50008 on yksisuuntainen tapahtumakanava, jonka kautta lukijan tapahtumia (events) voidaan vastaanottaa ulospäin. Lukijan event-tapahtumia vastaanottavat ohjelmat saavat oman kanavan ja kanavanumeron lukijalta. Event-kanavaa käyttämällä ohjelma voi rekisteröidä lukijan lähettämiä tapahtumia, kuten event.tag.arrive-lukutapahtumia. (Sirit Infinity 510 Protocol Reference Guide. 2009.)

Lukijan ja hallintaohjelman välinen kommunikointi vaiheittain:

1. Tapahtumakanava ja portti 50008 avautuu TCP-yhteyden takia lukijaan.
2. Ohjelma katsoo kanavanumeron.
3. Komentokanava ja portti 50007 avautuu TCP-yhteyden takia.

4. Kanavanumeron avulla kuunnellaan haluttuja tapahtumia.

5. Komentokanavan sulku ohjelman puolelta, mikäli ei käskyjä lukijalle.

(Sirit Infinity 510 Protocol Reference Guide. 2009.)

Lukijan virallisen RTS-ohjelmiston avulla käyttäjä pystyy hallitsemaan lukijaa graafisen käyttöliittymän avulla. Esimerkiksi lukijan sisäisen muistin hallinta on mahdollista. Lukijan sisäisessä muistissa on lukijan oma tietokanta. Lukijan sisäiseen muistiin mahtuvien tunnisteiden määrä on 10000 kpl. (Sirit Infinity 510 User's Guide. 2009.)

3 Langattomien tekniikoiden vertailu

Toimeksiantaja halusi yleis- ja hintavertailun eri langattomista tekniikoista, joita voisi hyödyntää kävijäseurannassa. Vertailtavat tekniikat olivat RFID, NFC ja iBeacon. Kysymyksiä, joihin toimeksiantajan puolelta haluttiin vastaus, olivat:

1. Järjestelmän hinta?
2. Järjestelmän koko?
3. Järjestelmän soveltuvuus?

RFID

Tutkittujen RFID-lukijoiden hinnat vaihtelivat välillä 445 - 1585 dollaria. Mitä kalliimpi laite, sitä enemmän ominaisuuksia se tarjosi. Yli 1000 laitteet vastasivat Sirit Infinity 510 –RFID-lukijaa. Lukija-antennit olivat hinnoiltaan 150 - 995. Tutkitut RFID-lukijat vastasivat kooltaan ja ominaisuuksiltaan Sirit Infinity 510 -RFID-lukijaa. (RFID-products. n.d.)

Tutkittujen RFID-tunnisteiden hinnat olivat välillä 25 - 300, lukumäärästä ja ominaisuuksista riippuen. Aktiivitunniste, joka on koteloitu, voi maksaa 100 dollaria kappaleelta. (RFID-products. n.d.)

Valmiin RFID-järjestelmän kokonaiskustannukset riippuvat täysin käytetyistä laitteistoista, niiden kokoluokasta ja käyttötarkoituksista. Tiedonvälitykseen ja hallinnointiin liittyy myös kustannuksia, esimerkiksi pilvipalveluihin ja sähkömaksuihin.

NFC

Tutkitut NFC-lukijat maksavat noin 30 - 70 dollaria. Tyypiltään ne ovat pienikokoisia pöytälukijoita, joita voi pitää myös kädessä. NFC-tunnisteet maksavat muutamista senteistä dollareihin riippuen tunnisteiden lukumäärästä ja valmistajasta sekä tilauspaikasta. Liitteessä 11 on esitetty NFC-pöytälukija ja NFC-tunniste. (NFC Reader. n.d.)

iBeacon

Tutkittujen iBeacon-laitteiden hinnat vaihtelevat välillä 15 - 100 dollaria. Hinta määräytyy valmistajan, mallin ja laitteiden lukumäärän mukaan. Tietoa iBeaconeista on vain vähän saatavilla. Alla tietoja eräästä iBeaconista (Roximity):

1. Bluetooth® Smart / Bluetooth Low Energy.
2. Ad rate: 100ms (iBeacon Certified).
3. Range: 60 meters / 200 feet.
4. Battery life: Up to 5 years.
5. Operating temp: -40 to +85 degrees Celsius.
6. Indoor/outdoor: NEMA 3R enclosure.
7. Security: Patent Pending ROXIMITY/iBeacon security.
8. Made in USA.

(Roximity iBeacon. n.d)

Yleisvertailu

RFID-tunniste voi olla passiivinen tunniste, joka aktivoituu vasta lukijan lähettyvillä (tapahtumakäyttöön sopiva). NFC-tunnisteen on oltava matkapuhelimessa kytkettynä päälle, ja tällöin se käyttää matkapuhelimen akkua. Salaus NFC:ssä on parempi, mutta toisaalta RFID-tiedot menevät tietokantaan käyttäjänimen ja salasanan taakse. Pankkipalveluiden hoitoon NFC olisi parempi vaihtoehto kuin RFID-tekniikka.

iBeacon tukee Applen laitteita, pääasiassa puhelimia. Kaikilla tapahtuman kävijöillä ei varmasti ole Applen puhelinta tai laitetta. iBeacon on sopiva suuremman tietomäärän siirtämiseen käyttäjän laitteeseen kuin laitteesta langattomaan lukijaan päin.

4 Tekninen toteutus

4.1 Työympäristö

Tekniikoiden vertailun tulosten esittämisen jälkeen toimeksiantajan kanssa pidettiin palaveri ja sovittiin opinnäytetyössä käytettävästä tekniikasta. Valinnaksi muodostui RFID. Tämän jälkeen ohjaavan opettajan kanssa sovittiin aikataulutuksesta ja luvasta käyttää koulun Sirit INfinity 510 RFID -laitteistoa opinnäytetyön tekemiseen. Työssä käytettiin JAMK:n Dynamolla sijaitsevaa RFID-kehikkoa ja kannettavaa tietokonetta tietokannan ja tietokantayhteyden muodostamiseen RFID-laitteiston kanssa. RFID-järjestelmä sijaitsi koulun luokassa D334, johon minulla oli pääsyoikeus. RFID-tunnisteina käytettiin RFID-luokan tuntiharjoituksesta tuttuja elintarvikkeita, joihin oli kiinnitetty passiivisia RFID-tunnistetarroja.

Jotta opinnäytetyötä pystyttiin tehdä myös etänä kotoa käsin, oli RFID-lukijaan pystyttävä luomaan etäyhteys. Ohjaavan opettajan kanssa mietittiin, miten etäyhteys olisi mahdollista toteuttaa:

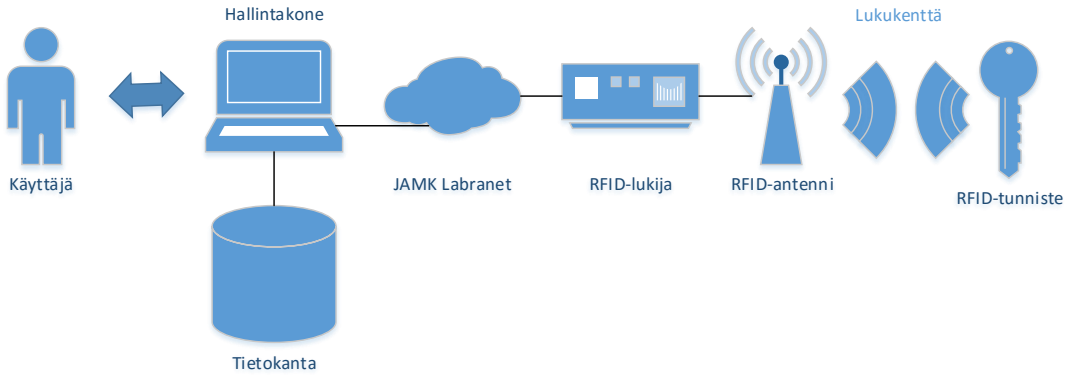
1. Remote Desktop -etäyhteys koulun RFID-luokkahuoneen tietokoneelle, jossa lukija on kiinni verkkokaapelilla.

2. VPN-tunneli JAMK Labranettiin ja lukija kiinteällä IP-osoitteella kiinni samassa Labranet-verkossa.

Opettajan ja JAMKin laboratorioinsinöörien kanssa päädyttiin ratkaisuun 2. Lukijalle varattiin sen MAC- eli rautaosoitteen 00:17:9E:00:10:18 perusteella Labranetin IP-osoitealueesta osoite 192.168.51.9, joka pysyisi varattuna ainoastaan lukijalle. Teknisen toteutuksen topologia on nähtävillä kuviosta 16.

Lukijaan muodostettiin hallintayhteys käyttämällä VPN-tunnelia Labranetiin. Lukijan pystyttiin todeta olevan verkossa kannettavan tietokoneen komentorivin komennolla *ping 192.68.51.1*, johon laite vastasi onnistuneesti. Komentorivin kautta lukijalle oli mahdollista syöttää komentoja ensin kirjautumalla laitteeseen *ssh cliuser* -

komennolla. Tällöin lukijaa päästiin hallitsemaan samalla tavalla kuin verkkopohjaisen CLI:n kautta.



Kuvio 16. Teknisen toteutuksen topologia

4.2 Infinity 510

Teknisen osan toteutus aloitettiin tutustumalla ensin Sirit INfinity 510 RFID -lukijaan. Lukija ja sen neljä antennia olivat kiinnitettyinä kehikkoon, rekkiin, jonka läpi kävely onnistui helposti. Lukijan manuaali ja protokollaopas löydettiin Internetistä, ja niihin tutustuttiin huolellisesti. Sen jälkeen etsittiin työn aiheeseen sopivista kirjoista ja internetistä vastaavanlaiseen RFID- järjestelmään liittyvää materiaalia ja perehdyttiin siihen. Teknistä osaa työstettäessä jouduttiin aika ajoin etsimään verkosta uusia ideoita ja ohjeita RFID:tä koskien, jotta työ etenisi.

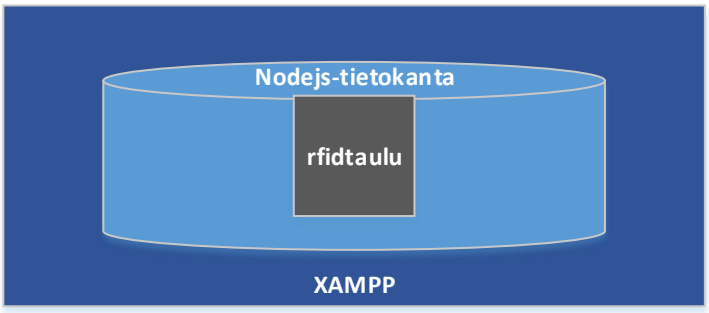
Lukijan hallintasovellus kopioitiin luokkahuoneen RFID-koneelta työssä käytettävälle kannettavalle tietokoneelle. Kannettavan verkkoasetukset tarkistettiin, ja ne olivat oletusasetuksilla. Kun verkkoasetukset olivat kunnossa ja IPv4-yhteys ylhäällä, lukijan pystyi näkemään hallintasovelluksessa. Komentoriviä käyttäen lukijan IP-osoitetta 192.168.51.9 pystyttin myös onnistuneesti pingaamaan. Etäyhteyttä varten kannettavan verkkoasetukset palautettiin alkuperäisiksi ja etäyhtedessä käytettiin VPN-yhteyttä.

Sirit RST -hallintasovelluksella pystyttiin säätämään lukijan eri ominaisuuksia, esimerkiksi antennien määrää, lukuetäisyyttä, tunnisteiden raportointia ja lukijaprofiileita.

Lukijan parametreja pystyttiin säätämään myös hallintaverkkosivun kautta. Tällöin selaimen osoiteriville tuli kirjoittaa lukijan IP-osoite ja painaa enter. Avautuvan virallisen hallintasivun kautta pystyttiin hienosäätämään lukijan asetuksia, päivittämään lukijan ohjelmistoversio, asentamaan lukijaan erilaisia Python- ja Java -skriptejä sekä hallita käyttäjäoikeuksia. Liitteessä 1 on kuvailtu lukijan alustuksen eri näkymät. Lukijan ja työaseman välisen yhteyden täytyi olla kunnossa ennen kuin siirryttiin seuraavaan työvaiheeseen, joka oli rajapinnan luominen tietokantayhteyttä varten.

4.3 XAMPP

Seuraavaksi asennettiin Xampp liitteen 2 mukaisesti. Työn toteutukseen tarvittiin vain Apache- ja MySQL-palvelut. Asennuksen jälkeen XAMPP:n asetuksia ei tarvittu muuttaa. Oletusasetukset toimivat oikein heti. Apache ja MySQL käynnistettiin ja niiden toimivuus testattiin menemällä selaimella osoitteeseen ”localhost”. Avautuva sivu oli XAMPP-aloitussivu. Samalle sivulle päästiin klikkaamalla XAMPP-hallintapaneelista Apachen kohdalta ”admin”-painiketta. Aloitussivun kautta olisi ollut mahdollista käyttää myös phpmyadmin-tietokantasovellusta. XAMPP-ympäristön hahmotelma on esitetty kuviossa 17.



Kuvio 17. XAMPP-alustan osat

4.4 MySQL

Tietokannan hallintaa varten asennettiin MySQL Workbench -sovellus. Se mahdollisti tietokantayhteyden luomisen ja tietokannan alustuksen. ”Tietokantarajapinnat”-kurssilla käytettiin samaa sovellusta, joten sen käyttö oli tuttua. Tietokantayhteyden asetukset määriteltiin liitteen 3 mukaisesti. Yhteyden nimeksi kirjoitettiin ”Oppari” ja

pääkäyttäjäksi valittiin ”root” ja pääkäyttäjän salasanaksi ”root”. Muut asetukset sovellus valitsi itse, jonka jälkeen asetukset hyväksyttiin. Tietokantayhteys testattiin toimivaksi, minkä jälkeen pystyttiin luomaan tietokanta tunnistetietoja varten.

Tietokanta luotiin liitteen 4 sisältämien SQL-komentojen mukaisesti Query-välilehdellä, jossa komennot pystyttiin ajamaan. Tietokannan olisi voinut luoda myös käyttämällä Workbenchin työkalupaneelin kautta avautuvaa näkymää, joka on esitetty liitteessä 5.

SQL-komentojen suorittamisen jälkeen tietokanta ilmestyi sovellukseen tarkasteltavaksi. Luotiin sarakkeet ”rfidkolumni”, ”rfidtime” ja ”rfidid”. Näiden sarakkeiden alle lukutapahtumasta tulee tieto tunnisteesta, tunnisteen lukuhetkestä ja tunnisteen järjestysnumerosta. Kolumni ”rfidkolumni” on muotoa VARCHAR ja voi sisältää 30 merkkiä. ”rfidtime” on DATETIME-muotoinen ja ”rfidid” on INT 11 merkkiä. Kolumneille määriteltiin attribuutteja niin, että mikään kolumni ei voi olla tyhjä (not null) ja kaikki kolumnien arvot ovat yksilöllisiä (unique). Tällä tavoin tietokantaan ei tule toistuvia arvoja samasta tunnisteesta. Jos sama tunniste tulee tietokantaan, ainoastaan sen aikaleiman arvo muuttuu.

4.5 Lukijaskripti

Tietokantayhteyttä varten hahmoteltiin PHP-skripti, joka alustaisi PHP-soketin ja ottaisi yhteyttä lukijaan. Tämän jälkeen skripti kuuntelisi lukijan event-kanavaa. Lukuprosessin käynnistyttyä lukukenttään saapuneesta tunnisteesta lukija syöttäisi event-kanavalle tunnistetiedon sisältävän event-viestin. Event-viestin sisältämä tieto siirtyisi tietokantaan skriptin avulla. PHP-skripti ei kuitenkaan toiminut, vaikka socket alustui onnistuneesti. Lukijan asetus event-viestien lähetykselle ei ollut kytkettynä päälle. Pitkän selvittelyn jälkeen päädyttiin johtopäätökseen, että parametria ei voitu kytkeä päälle tuntemattomasta syystä. PHP-komennoilla tunnistetiedon lisäys tietokantaan onnistui kuitenkin manuaalisesti.

Seuraavaksi testattiin lukijan sisällä ajettavan skriptin toimivuus. Lukijan muistissa sijaitsi malli Python-skriptistä, jota lukija ymmärtäisi. Python-skriptiä muokkaamalla testattiin lukuprosessin toimivuutta huonolla menestyksellä. Vaikka Python-skripti oli

lukijan muistissa ajossa, ei lukijan omaan tietokantaan eikä MySQL-tietokantaankaan tullut tietoa havaitusta tunnisteesta. Sisäisen skriptin toimimiseen luultavammin vaikutti myös lukijan event-asetusarvo ”pois päältä”.

Lukijaskriptin rakennetta tuli miettiä uudelleen. Se ei voisi vain kuunnella lukijaa, vaan sen tulisi pystyä käskemään lukijaa. Ongelman selvittelyn jälkeen päätettiin koettaa node.js-skriptiä lukijan komentamiseen. Node.js olisi kevyt prosessiltaan ja jäisi käynnistyksen jälkeen taustalle ajoon.

Lukijaskripti on sisällytetty liitteeseen 7. Skripti rakentuu muuttujien määritysosasta, toimenpideosasta, tietokantaosasta ja yhteyden katkaisuosasta. Muuttujia skriptissä ovat:

- Lukijan IP
- Lukijaportti
- Antenni
- Antennin käyttöteho
- Lukukomento
- Tietokantayhteysmääritykset

Lukijaskriptin käynnistyttyä tapahtuvat seuraavasssa järjestyksessä sen toimenpiteet:

1. Tietokantayhteyden määritys ja luominen
2. Yhteyden onnistumisesta ilmoittaminen
3. Tapauskohtaiset yhteydenkatkaisutoimenpiteet
4. Tunnistetiedon siistiminen ja SQL-komennon alustus
5. Lukijayhteyden muodostaminen
6. Antennin tehon määritys
7. Tunnistekysely 0.5 sekunnin välein

8. Havaitun tunnisteen kirjaus tietokantaan

9. Siirtyminen kohtaan 7.

10. Käyttäjästä tai vikatilanteesta johtuva yhteyden katkaisu.

Lukijaskripti määritettiin käyttämään vain yhtä lukijan neljästä antennista. Tämä tehtiin siksi, että etäyhteyden avulla työskenneltäessä ei olisi mahdollista viedä lukukenttään tunnisteita. Joten pöytätasolle, antennin 1 eteen sijoitettiin RFID-tunniste sekä viesti ”älä koske”.

4.6 Hallintasivut

Hallintasivut luotiin tietokannan tunnisteiden ja lukijaskriptin eli lukijan hallitsemista varten. Ne ovat HTML-kielen mukaiset, johon sisältöä tuottaa PHP-protokolla. PHP:n avulla muodostetaan rajapinta tietokannan ja hallintasivujen välille. PHP sopi protokollaksi hallintasivuille hyvin myös siksi, että toimeksiantaja oli sitä erikseen toivonut. Hallintasivujen rakenne on esitetty liitteissä 6 - 10. Kuviossa 18 on hahmoteltu käyttäjän ja RFID-järjestelmän hallintasuhteet.

Hallintasivujen tarkoitus on olla selkeät ja yksinkertaiset käyttää. Ne rakennettiin perustelementeiltään kevyiksi käyttäjälle. Käyttäjän on helppo ymmärtää sivujen rakenne. Hallintasivustolla pystyy helposti esittämään opinnäytetyön teknisen osan toteutumisen.

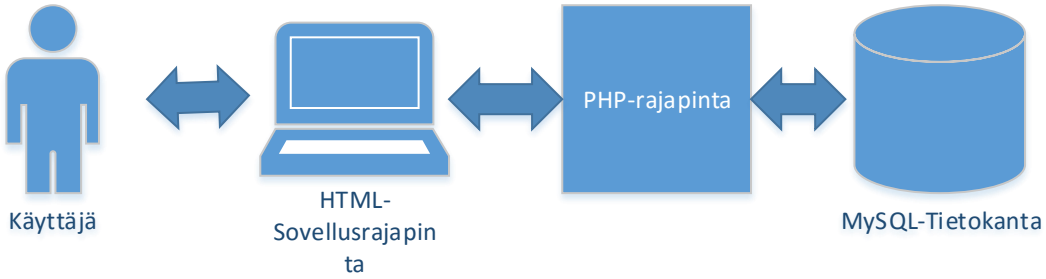
Hallintasivulle käyttäjä pääsee kirjoittamalla selaimeen osoitteeksi ”*localhost/rfid/RFID*”. Osoite on osa sivujen elementtien sijainnista XAMPP-sovelluksen kansiossa ”*C/xampp/htdocs/rfid/RFID*”.

Hallintasivut rakentuvat pääsivusta ja toimintosivuista. Pääsivulle on sijoitettu painikkeet RFID-järjestelmän eri toimintoihin. Ensimmäisestä painikkeesta ”Start” aukeaa sivu, joka käynnistää node.js-skriptin eli lukutapahtuman lukijassa. Skripti jää taustalle ajoon, vaikka käyttäjä palaa takaisin pääsivulle.

Seuraavasta painikkeesta ”Tagit” käyttäjä pystyy tarkastelemaan tietokannan sisältöä taulukkona esitettynä. Taulukkoon päivittyvät tietokannan tiedot sivua päivittämällä.

Kolmas painike ”Tyhjennä” käynnistää tietokannan sisältämän tiedon tyhjennyksen. Jos lukijaskripti on käynnissä ”Tyhjennä”-painiketta painettaessa, sen hetkiset tallennetut tiedot katoavat.

Viimeinen painike ”Stop” sulkee lukijaskriptin ja lukuprosessi pysähtyy. Sivun PHP-koodiin on upotettu komento, joka ajaa cmdclose.bat-tiedoston. Tiedosto sisältää komennon, joka sulkee komentorivi-ikkunan samalla pysäyttäen lukuprosessin.



Kuvio 18. Käyttäjän hallintasuhteet RFID-järjestelmässä

5 Mittaus

5.1 Mittauksen tavoitteet

Toimeksiantajan näkökulmasta haluttiin vastaukset neljään asiaan:

1. Järjestelmän käyttöönotto. Kuinka kauan laitteiston käynnistymiseen menee aikaa?

2. Tunnisteiden tallennusnopeus. Kuinka nopeasti tunnistetiedot siirtyvät tietokantaan tultuaan lukijan lukukenttään?

3. Tunnisteiden määrä. Kuinka monta tunnistetta voi lukea yhtä aikaa?

4. Tunnisteiden kantama. Mistä asti tunniste tunnistetaan?

5.2 Mittauksen suoritus

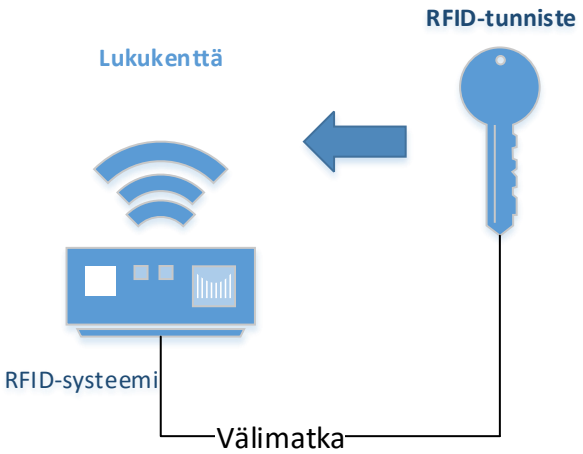
Mittaus suoritettiin JAMK:n Dynamon luokassa D334. Aluksi tarkistettiin, että RFID-järjestelmän kytkennät olivat tehty oikein. Hallintakone kytkettiin verkkovirtaan akun loppumisen estämiseksi. RFID-lukijan antennien kiinnitys tarkistettiin, samoin Ethernet-kaapelin kiinnitys. Mittauksen ajanottoon käytettiin älypuhelimen sekuntikelloa ja etäisyyden mittaamiseen mittanauhaa. Tulokset kirjattiin taulukkoon.

Ensimmäinen mittaus suoritettiin kahdessa osassa. Aluksi mitattiin hallintakoneen käynnistymiseen kulunut aika työpöydälle saakka. Hallintakone määritettiin käynnistymään suoraan työpöydälle, ja XAMPP-sovellus määritettiin käynnistymään automaattisesti. Apache ja MySQL käynnistyivät myös automaattisesti. Toiseksi mitattiin RFID-lukijan käynnistymistä lukijan virran kytkemisestä sen vastaamishetkeen. Asetettiin ICMP request → ICMP reply = 1 -rivi hallintakoneelle viivemittaukseen, jossa ping-komento ajettiin. Lukijan virta kytkettiin sen jälkeen päälle. Osatulokset yhdistettiin yhdeksi kokonaistulokseksi.

Toinen mittaus aloitettiin siten, että varmistettiin RFID-järjestelmän toimivuus sen käynnistyksen jälkeen. Toimivuus testattiin viemällä RFID-tunniste lukukenttään. Hallintasivulta nähtiin, että tunniste ilmestyi tietokantaan. Tämän jälkeen RFID-lukijaskripti käynnistettiin ”Start”-painikkeesta hallintasivulla. Tunniste asetettiin lukijan antennin eteen, ja se ilmestyi heti hallintasivun ”Tagit”-taulukkoon. Sama toistettiin viemällä 1 - 5 tunnistetta lukukenttään.

Kolmas mittaus suoritettiin asettamalla 10 kpl tunnisteita lukijan lukukenttään. Kaikki tunnisteet ilmestyivät tietokantaan välittömästi. Tunnisteita ei ollut saataville enempää kuin 10. Liitteessä 1 esitetyn lukijan alustuksen perusteella lukijan lukemien tunnisteiden määrän voi arvioida asetusparametrilla. Luettavien tunnisteiden määrä on välillä 1 - 500 tunnistetta. Voidaan olettaa, että lukijalla pystytään lukemaan yhtä aikaisesti lähemmäs 500 tunnistetta.

Neljäs mittaus suoritettiin tuomalla tunniste lukukentän läheisyyteen lukijan antenniin kohtisuorassa linjassa kuvion 19 osoittamalla tavalla. Samalla tarkkailtiin, milloin tunniste ilmestyisi tietokantaan. Tunnisteen tunnistushetkestä mitattiin välimatka tunnisteen ja lukijan antennin välillä. Lukijan antennin tehoa säätämällä lukukentän herkkyyttä pystyttiin mitoittamaan. Mitä pienempi teho (arvot 50, 100, 200 ja 300), sitä lähempänä tunniste oli antennia tunnistushetkellä.



Kuvio 19. RFID-tunnisteen vienti lukukenttään

5.3 Tulokset

Mittauksien tulokset kirjattiin taulukon 3 mukaisesti. Tuloksiin eivät vaikuttaneet ympäristötekijät, kuten toiset säteilylähteet tai luokan opiskelijat. Inhimillinen virhe on lähes olematon tarkkojen mittaskenaarioiden ansiosta. Tulokset olivat kaikissa mittauksissa yhtenevät mittauksien toistoista huolimatta.

Taulukko 3. Mittauksen tulokset.

Mittaus	Aika(s)	Määrä(kpl)	Etäisyys(cm)	Antenniteho(ddbm)
1.mittaus	**18(PC) + 30(lukija)=48**	**-**	**-**	**200**
2.mittaus	**< 1s**	**1 - 5**	**0**	**200**
3.mittaus	**< 1s**	**10**	**0**	**200**
4.mittaus	**-**	**1**	**15,30,60,90**	**50,100,200,300**

Työn tuloksista toimeksiantaja voi nähdä, miten työssä luotu RFID-järjestelmä käyttäytyy. Kävijäseurantaan Sirit-laitteisto sopisi hyvin nopeuden ja kapasiteetin osalta. Lukukentän mitoittaminen on varmasti positiivinen ominaisuus lukijan antennien sijoittelua ajatellen tapahtumapaikalla. Lukukentän mitoittaminen antaa vapautta mahdollisista ympäristörajoitteista. Esimerkiksi oviaukkojen ahtaus tai leveys pystytään ottamaan huomioon konfiguroitaessa laitteistoa. Tapahtuman kävijämäärää pystytään hallitsemaan mitoituksella. Esimerkiksi laaja lukukenttä ehkäisee jonotusruuhkan syntyä.

Tunnisteiden määrä jäi mittauksessa vähäiseksi, koska tunnisteita ei ollut enempää saatavilla. Onneksi tieto siitä, että lukija kuitenkin pystyy käsittelemään useita satoja tunnisteita tukee järjestelmän tapahtumakäyttötarkoitusta.

6 Analysointi

Mittausten tuloksista saatiin vastaukset toimeksiantajan esittämiin kysymyksiin. Laitteiston käyttöönottoaika on varsin nopea. Tieto nopeudesta antaa toimeksiantajalle mahdollisuuden aikatauluttaa järjestelmän käyttöönottoa tapahtumapaikalla. Laitteisto on myös vikasietoinen, jos lukijaprofiili on toiminnassa. Valmiiksi alustetut lukija-asetukset tallentuvat lukijaprofiiliin. Se tulee aina käyttöön lukijan uudelleenkäynnistyessä. Esimerkiksi sähkökatkon tapahtuessa lukijaa ei tarvitse konfiguroida uudelleen, vaan asetukset latautuvat käyttöön muistista. Toiminnallisuuden saavuttamiseen lukijan osalta riittää, että lukuprosessi käynnistetään hallintasivun kautta.

Laitteisto olisi myös tarpeeksi nopea sujuvaan tapahtumanhallintaan. Uusien kävijöiden saapuessa tapahtumapaikalle itse lukuprosessi ei aiheuttaisi turhaa viivettä. Kävijöille ei aiheutuisi myöskään turhaa odottelua lukuprosessin näkökulmasta. Kävijää ei tarvitsisi esimerkiksi kehottaa odottamaan lukuprosessin valmistumista ja tunnisteen tallennusta tietokantaan.

Tapahtuman kävijöiden määrä ei olisi myöskään ongelma tapahtumanhallinnan kannalta. Useita samanaikaisia kävijöitä pystyttäisiin tunnistamaan onnistuneesti ilman viivettä ja jonoa. Tunnistettavien kävijöiden määrä pystyttäisiin myös ennustamaan ilmoittautuneiden kävijöiden määrän avulla. Lukijalle olisi mahdollista tehdä tarvittavat säädöt kävijöiden lukumäärä huomioiden.

Lukukentän mitoitus on myös mahdollista säätämällä lukijaskriptin antennitehon arvoa. Jääkin toimeksiantajan päätettäväksi, miltä etäisyydeltä kävijä halutaan tunnistaa.

6.1 Johtopäätökset

Kehitetty RFID-järjestelmä on toimiva esimerkki. Se osoitti, että toimeksiantajan visio langattomasta tunnistusjärjestelmästä on mahdollista toteuttaa. Järjestelmä pystyi tarjoamaan vastaukset sille esitettyihin kysymyksiin varsin onnistuneesti.

RFID-järjestelmä on kallis, mutta pitkäikäinen kokonaisuus. Alkuinvestoinnin jälkeen lisäkustannuksia aiheuttavat tarvittavat RFID-tunnisteet. Järjestelmän ylläpidon pitäisi olla suhteellisen edullista, vaikka hallintaosuus sijaitsisi vuokratulla verkkopalvelimella.

6.2 Palaute

Toimeksiantaja oli tyytyväinen työn tuloksiin. Toimeksiantaja sai arvokasta materiaalia työn tuloksista ja pystyy halutessaan jatkokehittämään järjestelmää. JavaScriptin valjastaminen lukijan hallintaan oli toimeksiantajan kannalta hyvä asia, sillä tekninen osio pystyttiin näin todistamaan toimivaksi. JavaScript-osaamista toimeksiantajan yrityksessä on, joten JavaScript ei tuota ongelmia. Onnistuminen PHP-rajapinnan käytöstä hallintasivulla laskettiin myös positiiviseksi suoritukseksi.

Työn tietoperusta oli tarpeeksi kattava ja sisälsi opinnäytetyölle olennaisia asioita. Työympäristö oli toimeksiantajan mielestä hyvin toteutettu. Työn mittaskenaarion toteutus ja siitä saadut tulokset olivat kattavia ja palvelevat hyvin toimeksiantajan tuotteistus- ja kehitysprojektia.

7 Yhteenveto

7.1 Lopputulos

Aihe opinnäytetyölle syntyi tarpeesta kehittää uusi tapahtuman kävijätunnistusjärjestelmä. Opinnäytetyössä tutkittiin mahdollisia langattomia tekniikoita, jotka sopisivat opinnäytetyön tekniseen toteutukseen. Tutkituista tekniikoista päädyttiin käyttämään RFID-tekniikkaa kävijätunnistukseen. RFID-lukijan havaitsemat tunnistetiedot syötettiin automaattisesti niitä varten rakennettuun tietokantaan, jota ohjattiin juuri järjestelmää varten kehitetyllä hallintasivustolla. Työ saavutti sille asetetut vaatimukset, ja lopputulos oli onnistunut. Työn suoritus oli odotettua hankalampi ja aikaavievämpi prosessi.

Opinnäytetyön teoriaosuus päätettiin kirjoittaa ensin. Materiaalia löytyi koulun kurssimateriaaleista ja Internetistä. Teorioiden kattavuus vaihtelee aiheittain siten, että teknistä toteutusta vastaavat osiot ovat kattavimpia. Toteutus ja sen suoritus pyrittiin kuvailemaan mahdollisimman hyvin.

Opinnäytetyötä tehdessä vastaan tuli muutamia haasteita, mutta lopulta nekin ylitettiin. Suurin haaste oli RFID-lukijan lukijaskripti. Aikaisempaa JavaScript-tietämystä minulla ei ollut paljoakaan, joten sen opetteleminen vaati paljon pitkäjänteisyyttä. Lukijaskriptin kokoaminen ja testaus vaati myös paljon aikaa. Työn tekeminen helpottui huomattavasti, kun lukijaskripti saatiin toimimaan. Lopullinen RFID- järjestelmä pitää sisällään kaikenkaikkiaan noin 330 riviä koodia.

7.2 Jatkokehitys

Opinnäytetyöprosessin aikana toimeksiantajan kanssa pidettyjen palavereiden yhteydessä esiin tuli muutamia jatkokehitysideoita. Voitaisiinko RFID-järjestelmää esimerkiksi ohjata pilvipalvelusta? Olisiko se myös vikasietoinen vai ei? Aiheuttaisiko se monimutkaisempaa konfiguraatiota ja sitä kautta viivettä lukijaprosessiin? Onnistuisiko varavirtalähteen liittäminen osaksi järjestelmää? Miten sen pystyisi räätälöimään toimeksiantajan haluamalla tavalla?

Pohdin myös LBT-antennien käyttöä. Lukuprosessin keventäminen LBT-antennilla olisi ehdotonta, jos haluttaisiin lukuprosessin käynnistyvän vasta, kun tunniste saapuu lukukenttään. Tällöin lukukenttää ei tarvitse aktivoida 0.5 sekunnin välein kuten lukijaskripti nyt tekee.

Järjestelmän hallintasivut olisivat voineet olla ulkonäöltään edistyksellisemmät, mutta niiden tarkoitus olikin tässä työssä vain toiminnallisuuden todentaminen. PHP-koodiin olisi voinut pakottaa joidenkin sivujen automaattisen takaisinpaluun edelliseen sivuun. Kuitenkin nykyinen toteutus vaatii enemmän käyttäjän syötettä, mikä edistää järjestelmän toiminnallisuuden ymmärtämistä.

XAMPP-kehitysympäristön ja sen palvelut olisi voinut korvata node.js-pohjaisella palvelintoteutuksella. Tällöin teknisen toteutuksen koodi olisi ollut pääasiassa JavaScriptiä, eikä PHP:tä olisi tarvinnut käyttää lainkaan.

Työn tulosten luotettavuutta pystyttäisiin parantamaan isommalla mittausskenaariolla. Oikeita kävijöitä tunnisteiden kanssa ohjattaisiin lukukentän ohitse, eri aikaan ja eri etäisyyksiltä. Tälläisen mittauskenaarion toteuttaminen olisi ollut jo opinnäytetyön rajojen ulkopuolella.

7.3 Pohdinta

Opinnäytetyön työstäminen oli palkitseva prosessi. Etenkin lukijaskriptin työstäminen oli loppujen lopuksi hieno kokemus. Alkuperäisen lukijaskriptin epäonnistuminen ja uuden skriptin valmistuminen ja sen testauksen onnistuminen nopeutti muun toteutuksen valmistumista huomattavasti. Yllättäen tietokannan säätäminen oli muistia virkistävä vaihe. Viimeksi kolmantena opiskeluvuonna olin tietokannan parissa työskennellyt. PHP oli myös vanha tuttu aiemmista opinnoista.

Langattomiin tekniikoihin tutustuminen oli mielenkiintoista ja vielä palkitsevampaa oli kyky kertoa niistä toimeksiantajalle. Koetin rajata työtä toimeksiantajan kanssa sopimalla tutkittavien langattomien tekniikoiden määrän kolmeen. Näin työn teoriaosuus ei paisunut liikaa mutta työstä oli mahdollista saada riittävän kattava.

8 Lähteet

Bluetooth Basics. N.d. Artikkeli Bluetoothin virallisella sivustolla. Viitattu 20.6.2015. http://www.bluetooth.com/Pages/Basics.aspx.

Chapman, S. N.d. What is JavaScript? Artikkeli verkkosivulla. Viitattu 24.10.2015. http://javascript.about.com/od/reference/p/javascript.htm.

General Python FAQ. N.d. Tietosivu Python-kielestä. Viitattu 4.9.2015. https://docs.python.org/2/faq/general.html.

Getting Started with iBeacon. 2014. iBeacon-opas kehittäjän verkkosivulla. Viitattu 20.6.2015. https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf.

Häkkinen, A. 2013. Tietokantarajapinnat. Jyväskylän ammattikorkeakoulun Optima-opiskeluympäristön kalvosarja. Viitattu 5.10.2015. https://optima.jamk.fi.

Jaakkola & Sarja. 2006. Mikä on tietokanta? Artikkeli verkkosivustolla. Viitattu 5.10.2015. http://verkkopedagogi.net/vanhat/fi/sisalto/materiaalit/access2003/luku021c5a.html?C:D=419700&selres=419700.

JavaScript-perusopas. 2007. Opas JavaScriptin käytöstä. Viitattu 24.10.2015. http://www.ohjelmointiputka.net/oppaat/opas.php?tunnus=js_01.

Kilpeläinen, A. 2007. Relaatiotietokanta. Tiedote verkkosivustolla. Viitattu 5.10.2015. http://homes.jamk.fi/~kivni/http0140/material/relaatiotietokanta.html.

Mikä on PHP? N.d. PHP-opas verkkosivustolla. Viitattu 4.9.2015. http://hallstrom.fi/php_opas/php_1.html.

Near Field Communication. N.d. Virallinen NFC-sivusto. Viitattu 5.10.2015. http://www.nearfieldcommunication.org/.

NFC Frequently Asked Questions. N.d. Kysymyksiä ja vastauksia NFC-tekniikasta. Viitattu 10.9.2015. http://www.smartcardalliance.org/publications-nfc-frequently-asked-questions/#6.

NFC Reader. N.d. Verkkokaupan sivusto. Viitattu 19.3.2015. http://www.alibaba.com/showroom/nfc-reader-price.html.

NFC Tag Types. 2011. Lista tunnistetyypeistä verkkosivustolla. Viitattu 11.9.2015. http://www.nfc.cc/technology/nfc-tag-types/.

Node.js. N.d. Node.js-verkkosivu. Viitattu 12.10.2015. https://nodejs.org/en/.

Overview of Bluetooth Pairing. 2008. Tietosivu Microsoftin verkkosivustolla. Viitattu 23.102015. https://msdn.microsoft.com/en-us/library/cc510479.aspx.

Poole, I. N.d. RFID-Standards. RFID-standardeista kertova tietosivu. Viitattu 10.10.2015. http://www.radio-electronics.com/info/wireless/radio-frequency-identification-rfid/iso-epcglobal-iec-standards.php.

Prospectum Oy. N.d. Toimeksiantajan kotisivut. Viitattu 3.6.2015. www.prospectum.fi.

Relaatiotietokantojen peruskäsitteet. 2004. Jyväskylän avoimen yliopiston ohjesivu. Viitattu 5.10.2015. http://appro.mit.jyu.fi/doc/tiedonhallinta/tietokannat/index2.html.

RFID FAQ. N.d. Vastauksia RFID-Journal -verkkosivustolla. Viitattu 20.6.2015. http://www.rfidjournal.com/faq/show?49.

RFID frequency ranges. N.d. Ranskan kansallinen RFID-keskus -verkkosivu. Viitattu 24.10.2015. http://www.centrenational-rfid.com/rfid-frequency-ranges-article-16-gb-ruid-202.html.

RFID-lukijat. N.d. Tiedote verkkosivulla. Viitattu 4.9.2015. http://www.finn-id.fi/tuotteet/laitteet/rfid-lukijat.

RFID-products. N.d. Verkkokaupan sivusto. Viitattu 19.3.2015. http://www.atlasrfidstore.com/rfid-readers/?sort=bestselling.

RFID-standardit. N.d. Artikkeli RFIDlab Finland –verkkosivustolla. Viitattu 10.10.2015. http://www.rfidlab.fi/rfid-standardit.

Roximity iBeacon. N.d. Verkkokaupan sivusto. Viitattu 19.3.2015. http://buyibeacons.com/.

Sarja, J. 2006. Relaatiotietokanta. Artikkeli verkkosivustolla. Viitattu 5.10.2015. http://verkkopedagogi.net/vanhat/fi/sisalto/materiaalit/access2003/luku0375c6.html?C:D=419702&selres=419702.

Sirit Infinity 510 Protocol Reference Guide. 2009. 510-lukijan manuaali. Viitattu 26.5.2015. http://www.finn-id.fi/sites/default/files/infinity_510_prot_ref_gd_v3.0.pdf.

Sirit Infinity 510 User's Guide. 2009. 510-lukijan käyttäjän opas. Viitattu 26.5.2015. http://www.finn-id.fi/sites/default/files/infinity_510_users_guide_v3.0.pdf.

SQL-alkeet. N.d. Johdatus SQL:n maailmaan. Viitattu 20.6.2015. http://www.2kmediat.com/sql/alkeet.asp.

Thrasher, J. 2013. RFID vs. NFC. Artikkeli verkkosivustolla. Viitattu 15.7.2015. http://blog.atlasrfidstore.com/rfid-vs-nfc.

Violino, B. 2005. Basics of RFID. Artikkeli RFID-Journal -verkkosivustolla. Viitattu 20.6.2015. http://www.rfidjournal.com/articles/pdf?1337.

What is MySQL? N.d. Ohje MySQL.com –sivustolla. Viitattu 14.10.2015. https://dev.mysql.com/doc/refman/5.0/en/what-is-mysql.html.

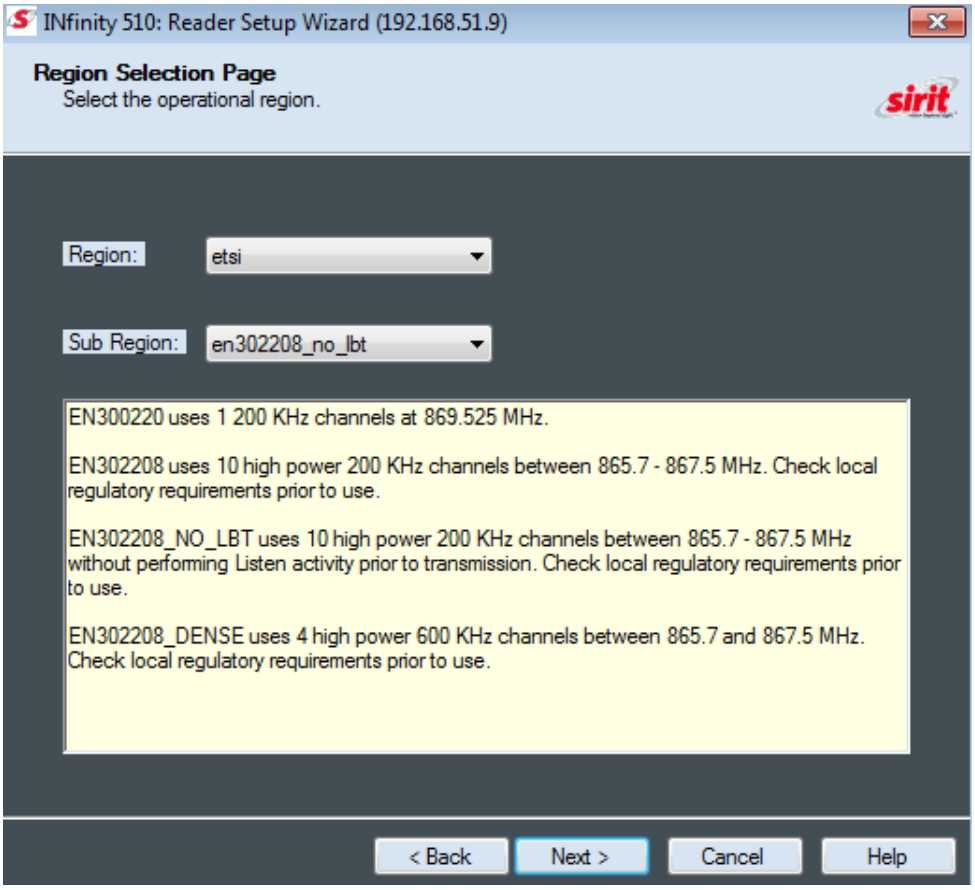
What is XAMPP? N.d. Selitys Xampp-kotisivulla. 14.10.2015. https://www.apachefriends.org/index.html.

Liitteet

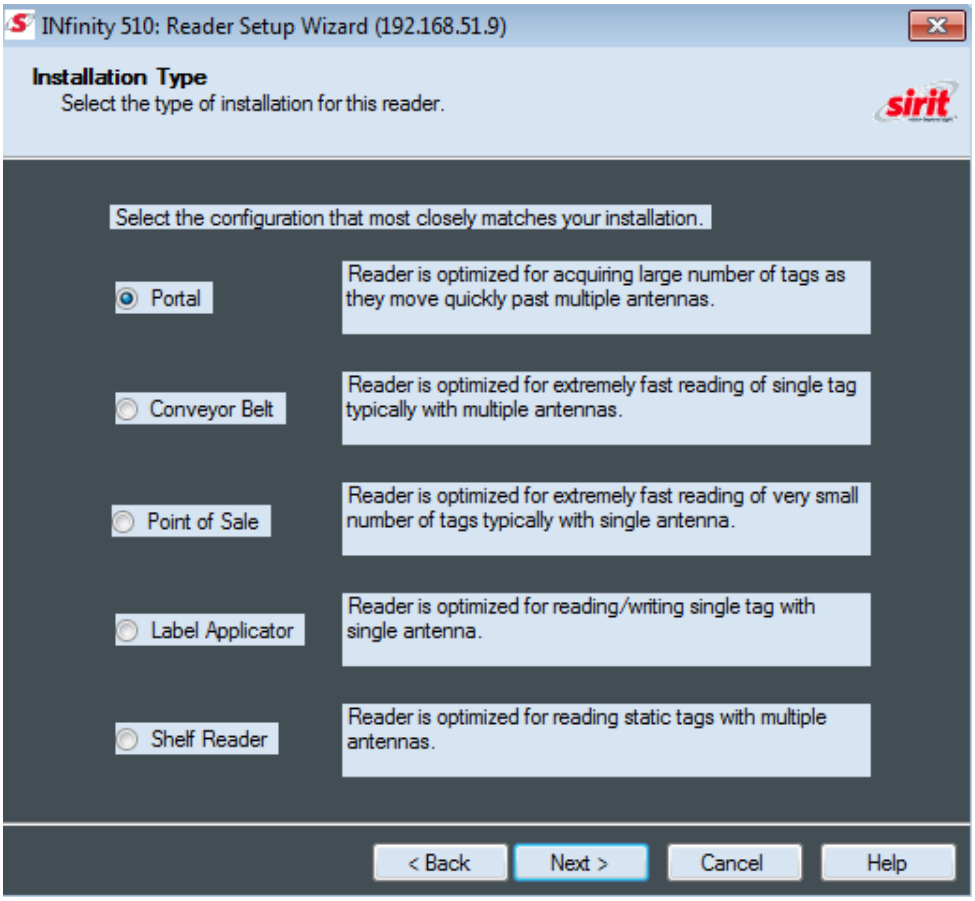
Liite 1 Sirit-hallintasovellus ja lukijan alkuasetukset

Reader Setup Wizard –alustusohjelman aloitusnäkymä.

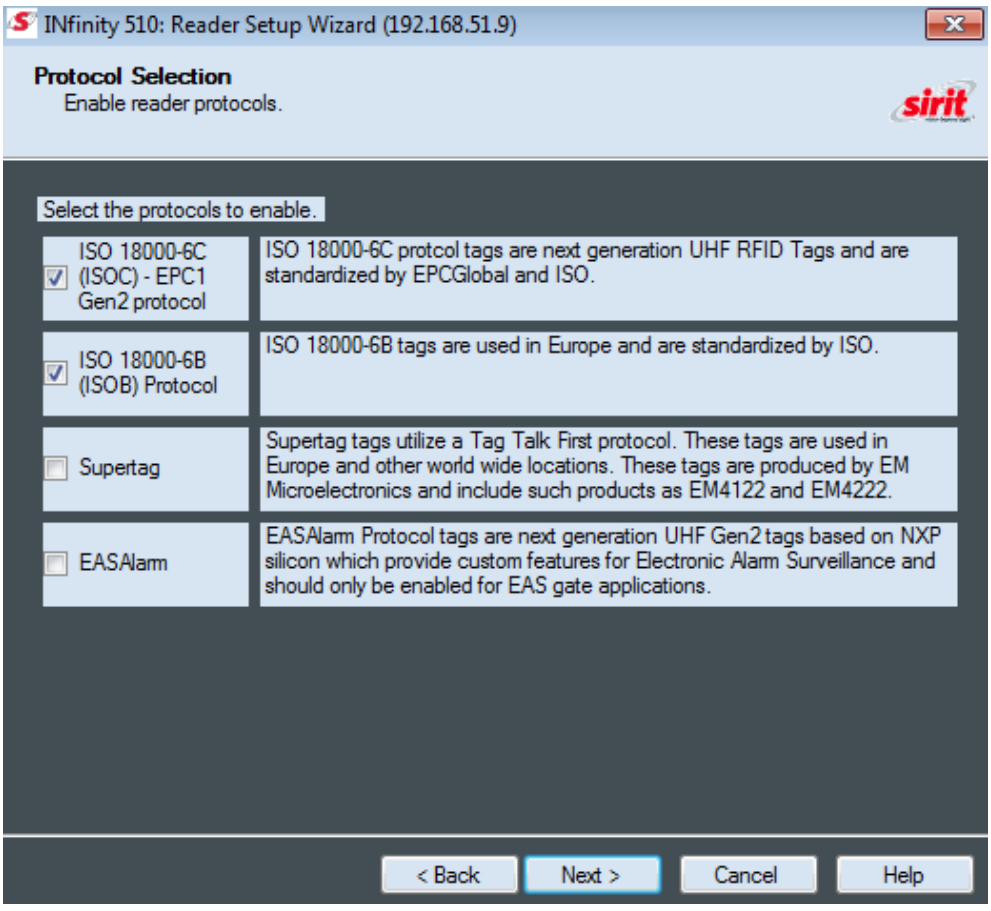
Lukija alustetaan käyttäen Sirit Infinity 510 Reader Setup Wizard -asennusohjelmaa. Se on osa Reader Startup Tool -ohjelmistoa. Ensimmäisestä ruudusta siirrytään eteenpäin next-painikkeella.



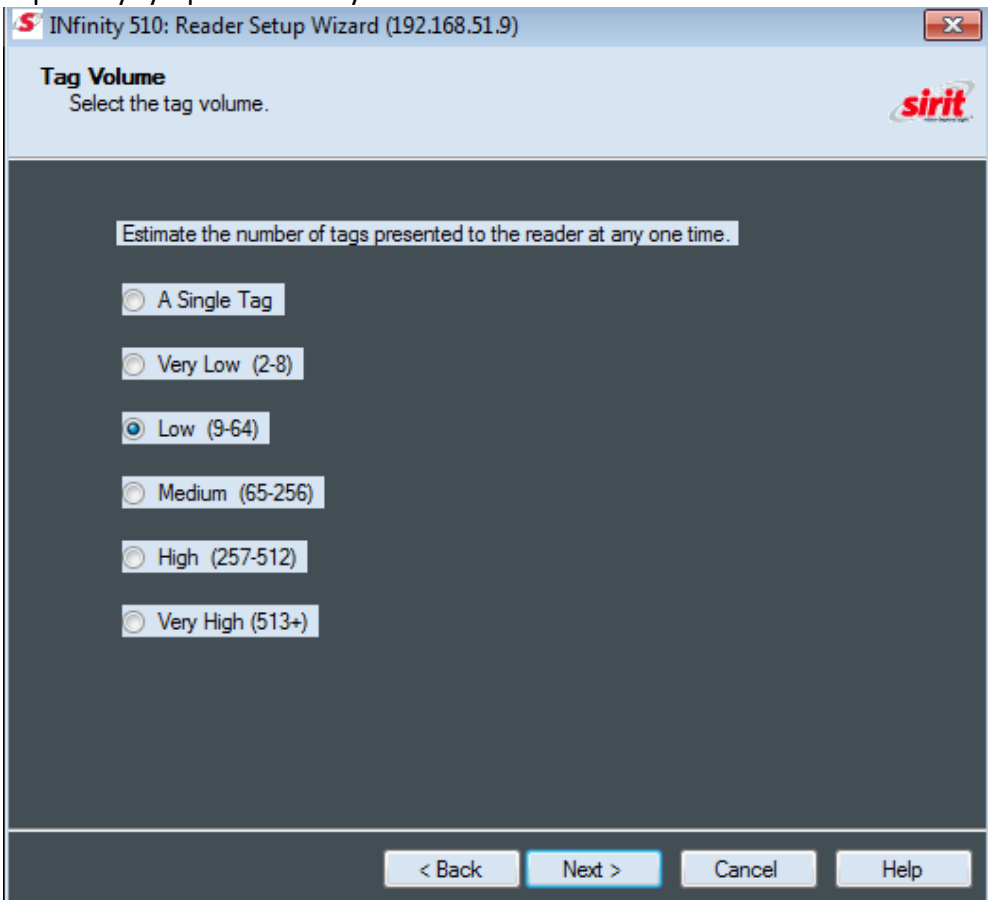
Lukijan toiminta-alueen valitseminen. Toiminta-alueen ”etsi” alta löytyy alavalikko, jonka arvoksi valitaan ”en302208_no_lbt”. Arvo mahdollistaa 10 kanavan käytön taajuusalueelta 865.7 - 867.5 MHz. LBT-antennia ei käytetä.



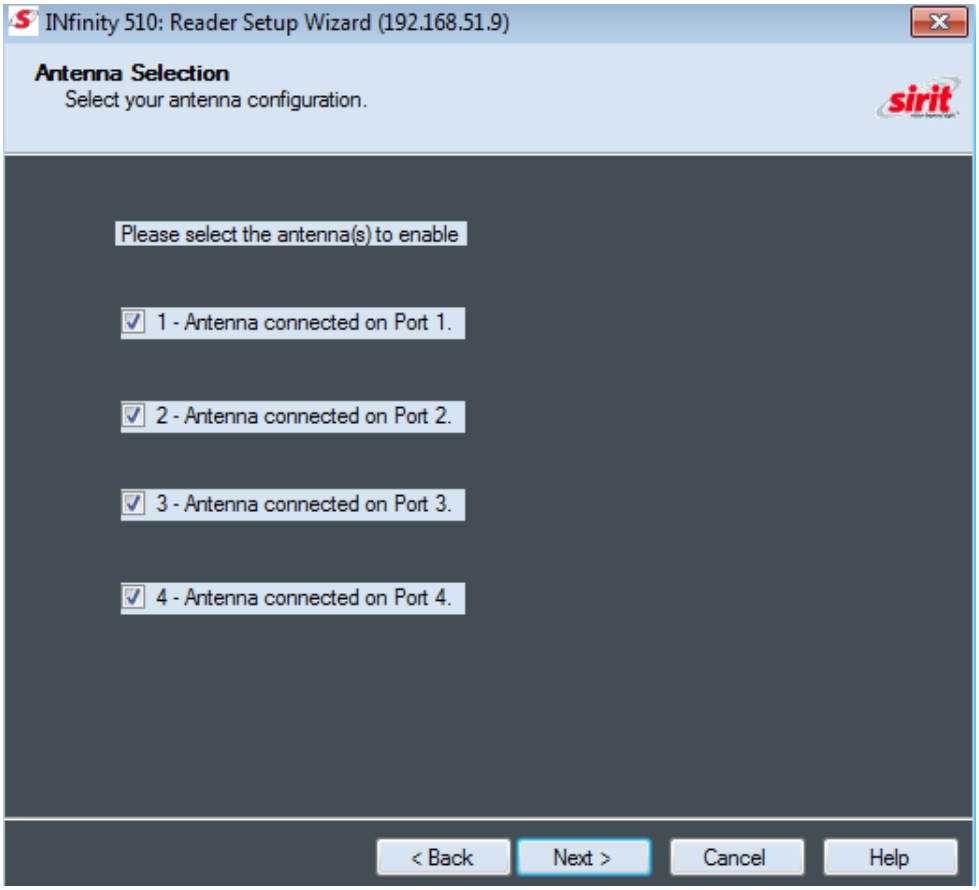
Lukijan toimintatavan valitseminen. Valitaan toimintatavaksi Portal eli porttikäytävä. Muita toimintatapoja ovat liukuhihna, lukupiste, yhden antennin ja monen antennin luku.



Lukijan protokollien valitseminen. Molemmat ISO 18000-standardit valitaan. Ne sopivat työympäristöön käytettäväksi.

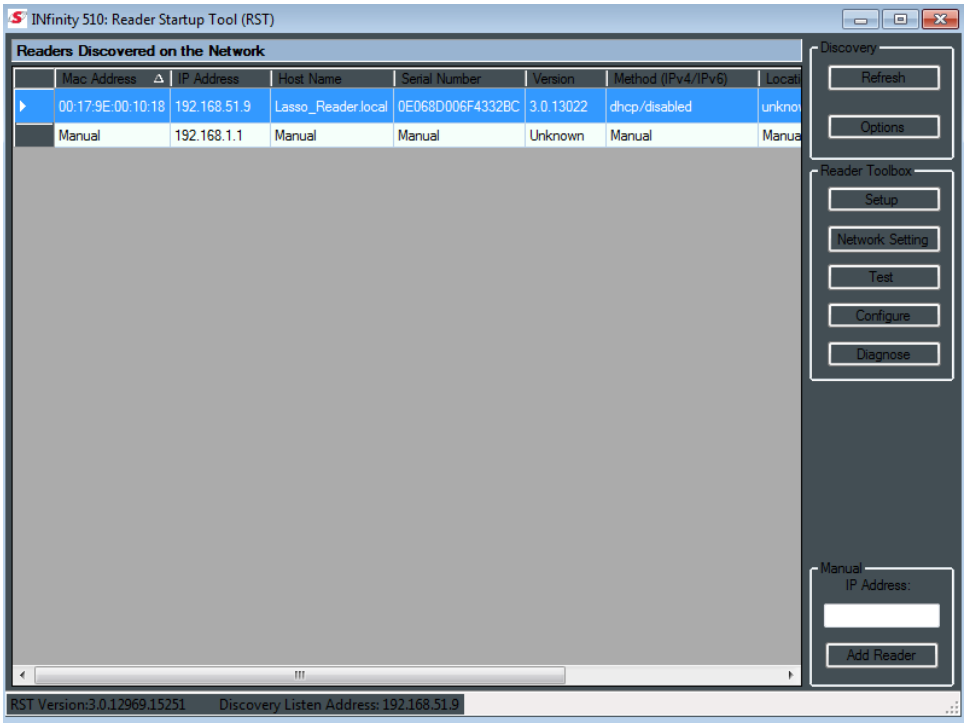


Lukijan tunnisteiden määrän valitseminen. Työympäristön rajoitteiden mukaisesti valitaan asetus ”Low (9 - 64)”. Tällöin lukija tietää, että lukukenttään mahdollisesti tulee 9 - 64 tunnistetta.

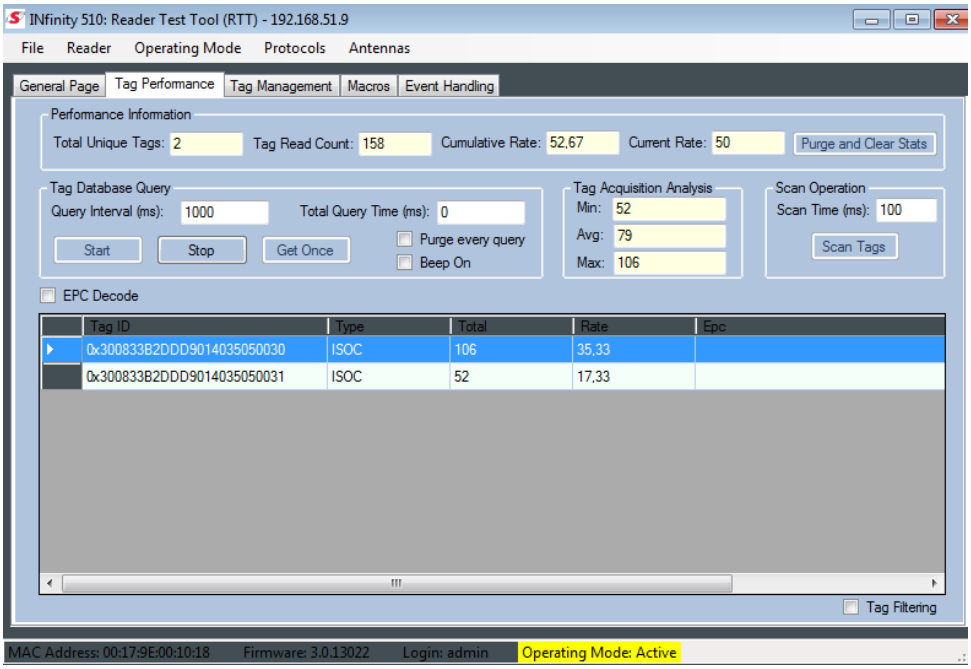


Lukijan antennien valitseminen. Lukijan antennien valinta. Alustuksessa valittiin käytettäväksi kaikki lukijan neljä antennia. Lukijaskripti kuitenkin käyttää vain ensimmäistä antennia.

Lukijaprofiilille annetaan nimi ja lukijaprofiili tallennetaan lukijan muistiin painamalla ”Finish”-painiketta.



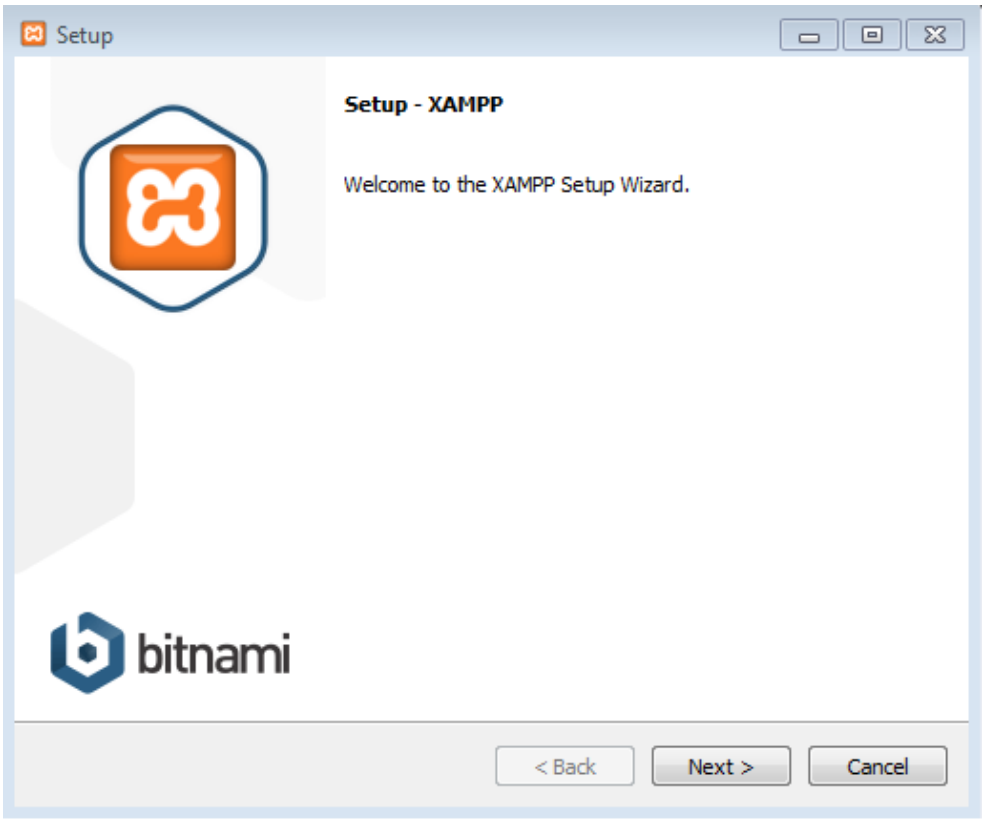
RTS-sovelluksen alkunäkymä alustuksen jälkeen.



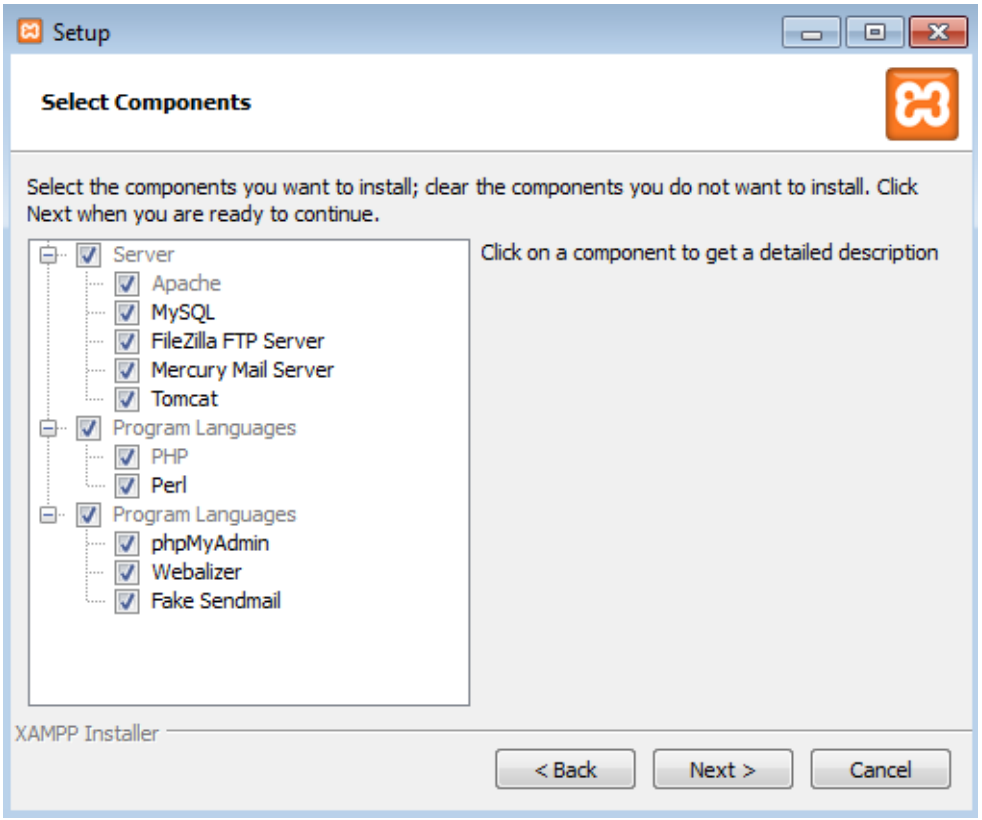
Lukijan lukukentän testityökalu. Käynnistymisen jälkeen lukukentässä havaitut tunnisteet näytetään hallintasovelluksessa.

Verkkopohjaisen virallisen hallintasivun ensinäkymä. Selaimella on menty lukijan IP-osoitteeseen 192.168.51.9.

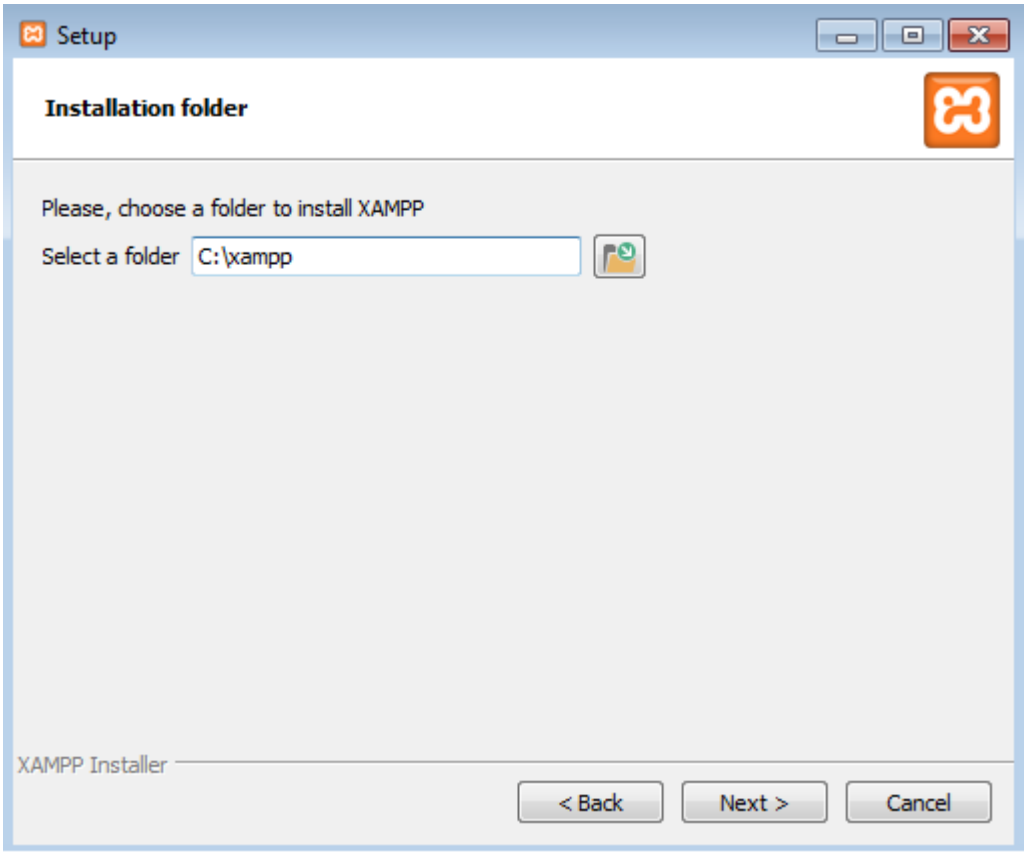
Liite 2 XAMPP-asennus



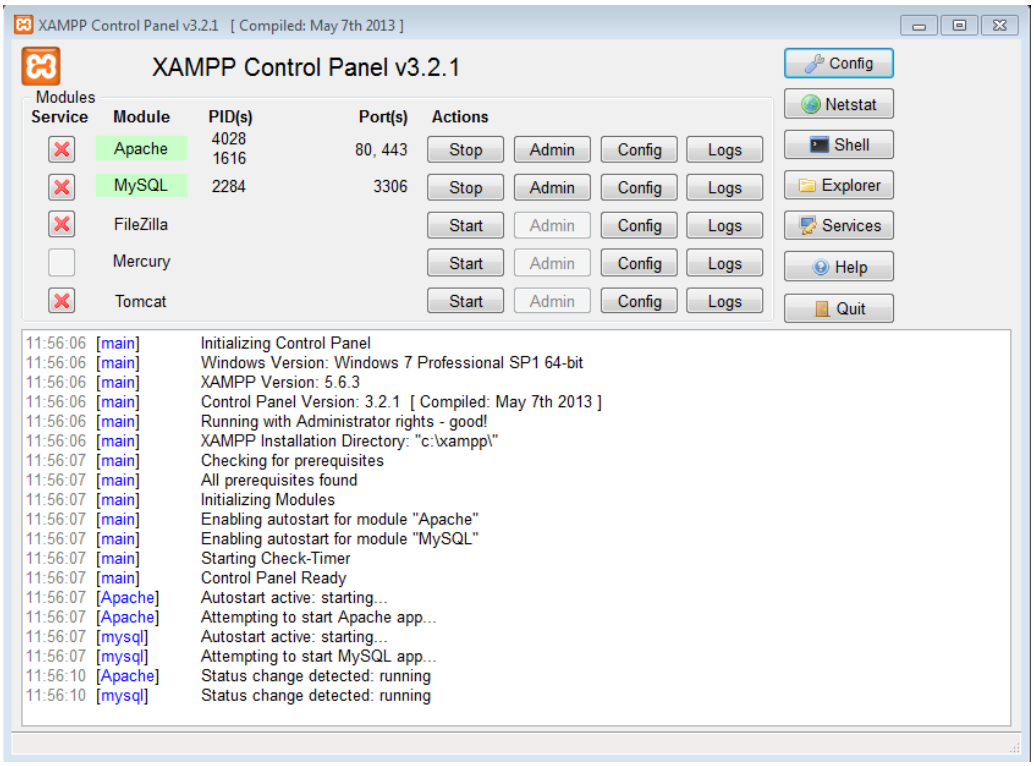
XAMPP-asennusohjelman aloitusikkuna. Eteenpäin asennusohjelmassa pääsee painamalla ”next”-painiketta.



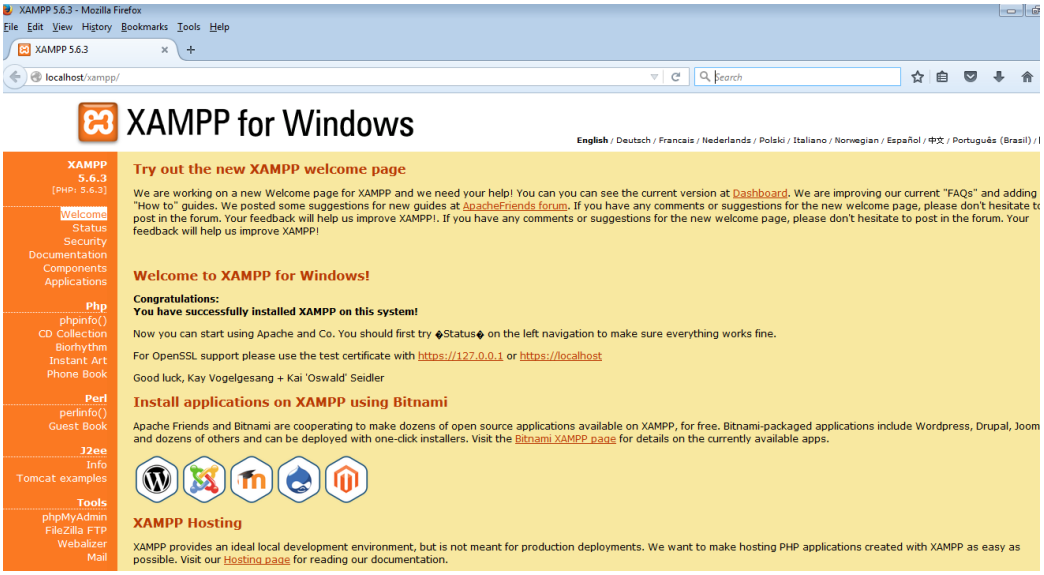
Asennettavien ohjelmien valitseminen. Kaikki tarjotut ominaisuudet voidaan asentaa, vaikkei niitä kaikkia käytettäisi.



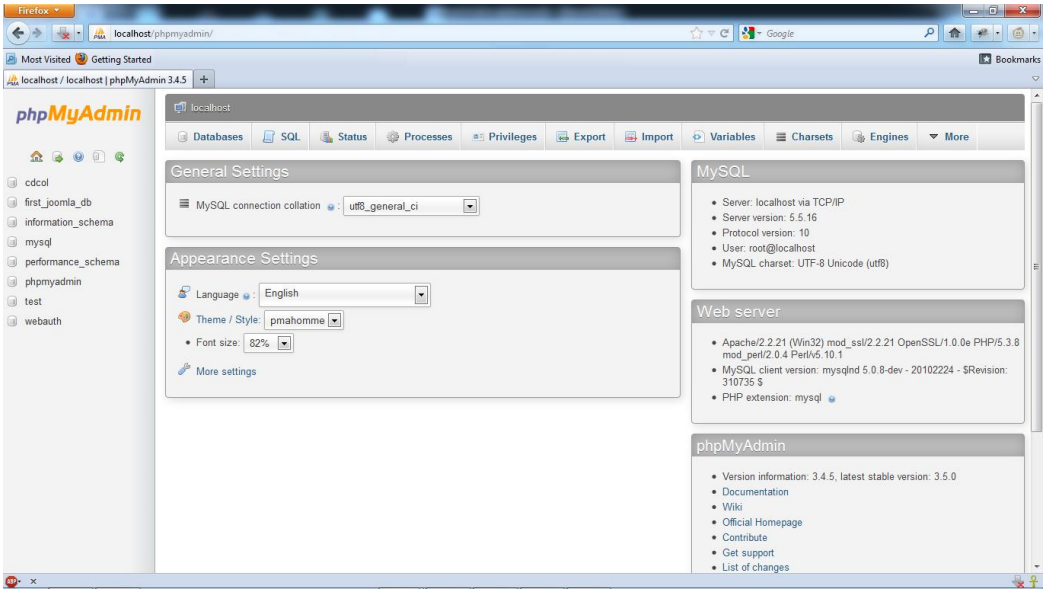
Asennuskansion valitseminen. XAMPP-asennetaan oletuskansioon ”C:\xampp”.



Asennuksen jälkeen avautuva XAMMP-ohjauspaneeli. XAMMP:n eri osia voidaan hallita niille määrätyistä painikkeista.

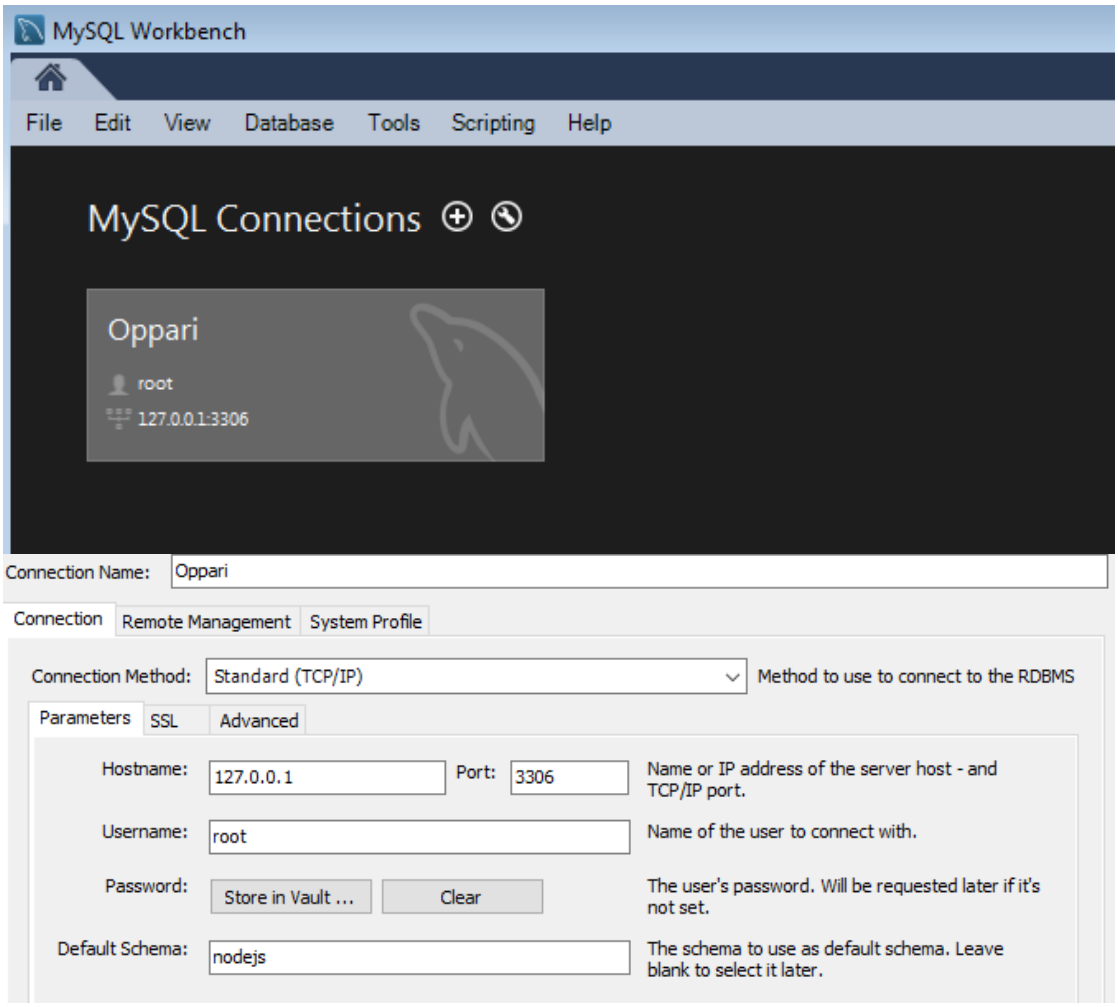


Localhost-sivun näkymä. Käyttäjälle ensimmäinen näkymä ”localhost”-verkkosivulle siirryttäessä.

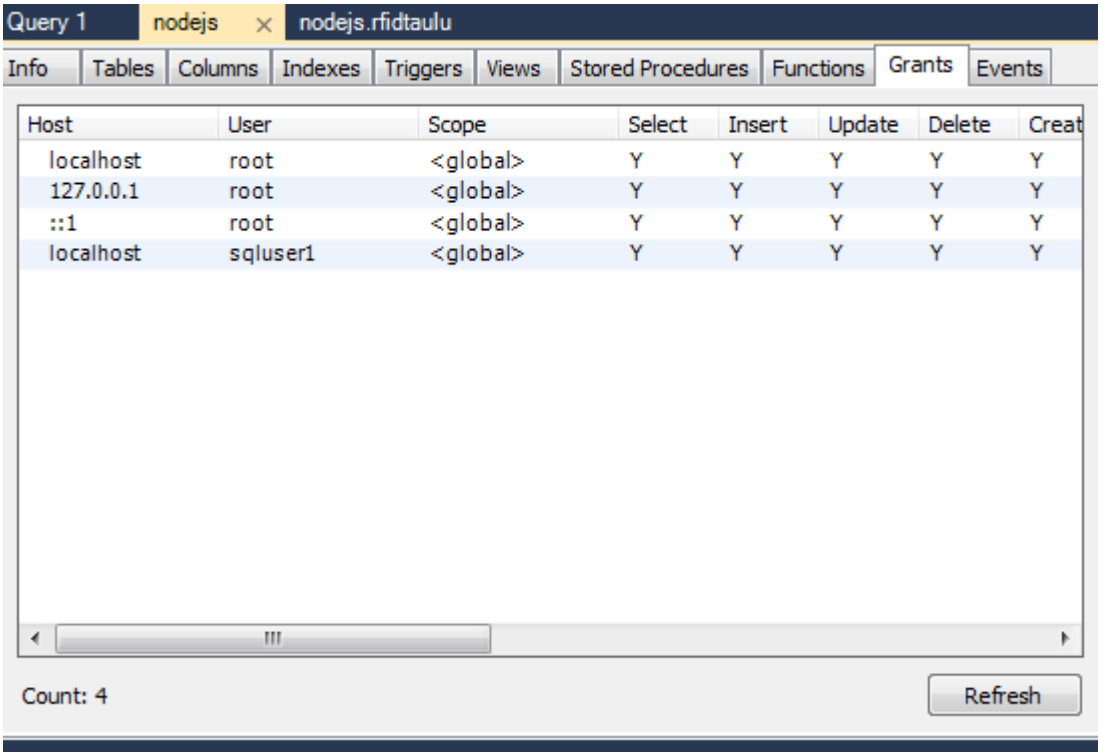


phpmyadmin-työkalun näkymä.

Liite 3 MySQL-tietokanta



Tietokantayhteys ja sen parametrit. Yhteysparametrit määriteltiin ja yhteys luotiin. Se ilmestyi valittavaksi yhteydeksi ”MySQL Connections” -tekstin alle.

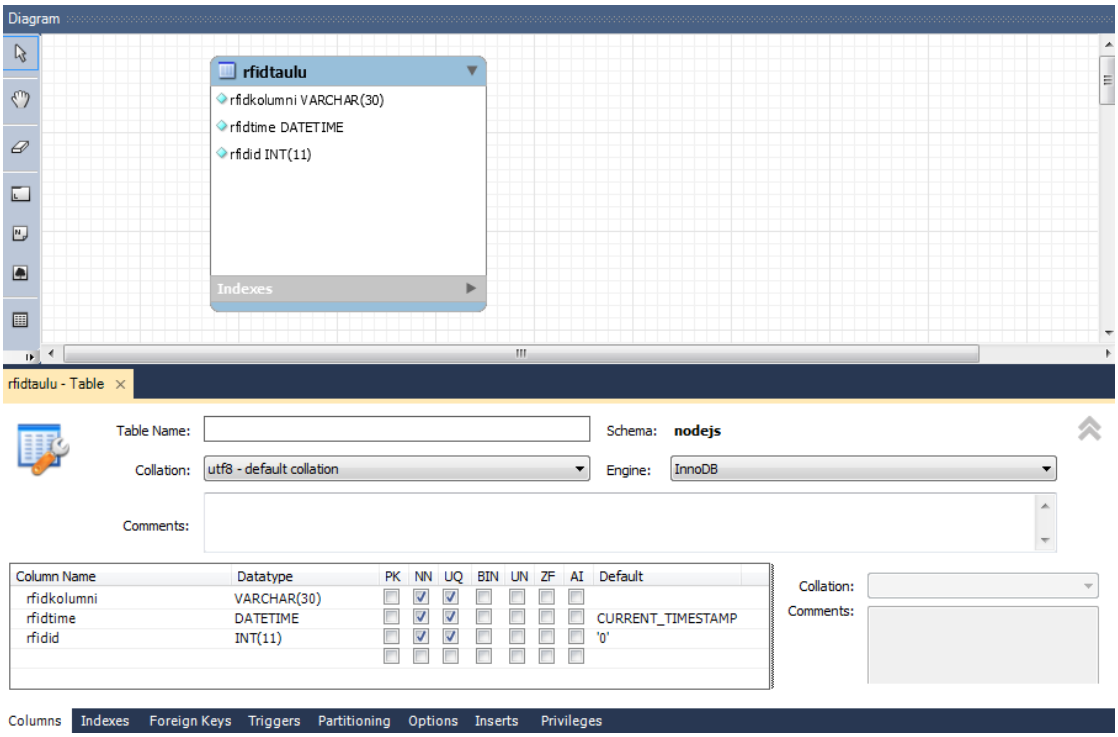


Tietokannan käyttäjät ja oikeudet. ”Nodejs”-tietokantaan on eri käyttäjillä on kaikilla samat täydet käyttöoikeudet.

Liite 4 MySQL-tietokantataulu

```
CREATE TABLE `rfidtaulu` (
`rfidkolumni` varchar(30) NOT NULL,
`rfidtime` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
`rfidid` int(11) NOT NULL,
UNIQUE KEY `rfidkolumni_UNIQUE` (`rfidkolumni`),
UNIQUE KEY `rfidtime_UNIQUE` (`rfidtime`),
UNIQUE KEY `rfidid_UNIQUE` (`rfidid`)\n)
ENGINE=InnoDB DEFAULT CHARSET=utf8'
```

Liite 5 Tietokannan kaaviokuvio



”Rfidtaulu”-taulun kaavionäkymä lisätietoineen.

Liite 6 RFID-hallintasivu

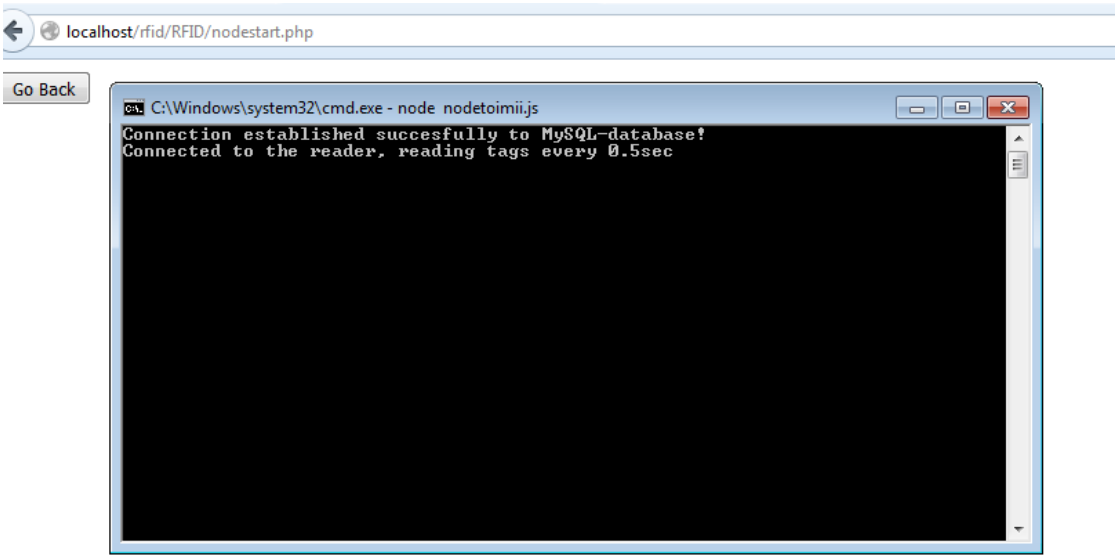
RFID-järjestelmän etusivu. Hallintasivun osoite selaimessa on ”localhost/rfid/RFID.

Lähdekoodi index.html

```
<!DOCTYPE html"
<html xml:lang="en" lang="en">
<head>
<title>RFID-hallintasivu </title>
<meta http-equiv="content-type" content="text/html;charset=utf-8" />
<link rel="stylesheet" type="text/css" href="rfid.css" />
</head>
<body>
<div id="mainContentArea">
<div id="contentBox">
<div id="title">RFID</div>
<div id="linkGroup">
<div class="link"><a href="nodestart.php">Start</a></div>
<div class="link"><a href="tagisivu.php">Tagit</a></div>
<div class="link"><a href="tyhjenna.php">DB-tyhjennys</a></div>
<div class="link"><a href="stop.php">Stop</a></div>
</div>
<div id="rfid">
<div id="header"></div>
<div class="contentTitle">RFID-hallintasivu</div>
<div class="pageContent">
<p>Tällä sivulla voit hallita RFID-tapahtumia.</p>
<p>1.Start-painike käynnistää tagilukuskriptin.</p>
<p>2.Tagit-painike näyttää tagitietokannan sisällön.</p>
<p>3.DB-tyhjennys-painike tyhjentää tagitietokannan. </p>
<p>4.Stop-painike pysäyttää tagilukuskriptin</p>
<p></p>
<p>Ongelmatilanteissa ota yhteyttä järjestelmänvalvojaan!</p>
</div>
<div id="footer">> </div>
</div>
```

```
</div>
</div>
</body>
</html>
```

Liite 7 Lukijan käynnistys ja lukijaskripti



RFID-lukijan käynnistyssivu. Avautunut komentorivi näyttää lukijaskriptin toimintoja.

Lähdekoodi nodestart.php

```
<!DOCTYPE html>
<html>
<body>
<?php
execInBackground('start cmd.exe @cmd /k "node nodetoimii.js"');
function execInBackground($cmd) {
   if (substr(php_uname(), 0, 7) == "Windows"){
      pclose(popen("start /B ". $cmd, "r"));
   }
   else {
      exec($cmd . " > /dev/null &");
   }
}
?>
 <button onclick="goBack()">Go Back</button>
<script>
function goBack() {
   window.history.back();
}
</script>
</body>
</html>
```

Javascript nodetoimii.js

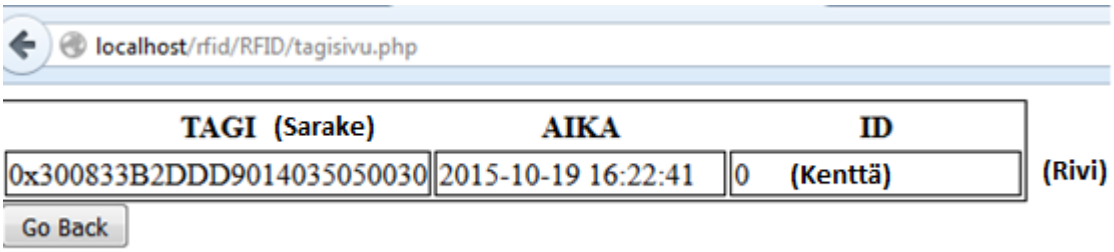
```
var net = require('net'); //*Käytettävien moduulien valinta*//
   var fs = require('fs');

   /*Configuration variables start=muuttujien määritys*/
```

```
// Use DEBUG-output instead of the MySQL. set to true or false.
var debug_instead_of_mysql = false;

// 1. Sirit INfinity device reader IP address.
var device_ip = '192.168.51.9';
// 2.Sirit INfinity device reader command channel port.
var device_port = 50007;
//  3.Use the antenna number 1.
var antenna_id = 1;
//  4.Command for setting more power to antenna.
var antenna_power_string = "antennas." + antenna_id + ".conducted_power = 250\r\n";
//  5.Command for reading the tag.
var command_string = "tag.read_id(" + antenna_id + ")\r\n";
// 6.MySQL-server hostname.
var mysql_host = 'localhost';
// 6.MySQL-server username.
var mysql_user = 'root';
// 6.MySQL-server password.
var mysql_pass = 'root';

// 6.MySQL-server default database.
var mysql_database = 'nodejs';

/*Configuration variables end*/

/*Operations start=skriptin toimenpiteet*/

var mysql;
var dbconnection;
if (!debug_instead_of_mysql) {
    // Load the MySQL-connector.
    var mysql = require('mysql'); //*Moduulin valinta yhteysmääritysten yhteydessä*//

    // 7. Setup MySQL database connect configurations.
    var dbconnection = mysql.createConnection({
        host: mysql_host,
        user: mysql_user,
        password: mysql_pass,
        database: mysql_database
    });

    // 8. Connect to the MySQL-database.
    dbconnection.connect(function(err) {
        if (err) {
            console.log('Error connecting to Db');
            process.exit(1);
        }
```

```
        console.log('Connection established succesfully to MySQL-database!');
    });

}
            // 9.Connection closing
            process.stdin.resume();//so the program will not close instantly
            function exitHandler(options, err) {
if (options.cleanup) console.log('dbconnection end');
if (err) console.log(err.stack);
if (options.exit) process.exit();
}
            //do something when app is closing
            process.on('exit', exitHandler.bind(null,{cleanup:true}));
            //catches ctrl+c event
            process.on('SIGINT', exitHandler.bind(null, {exit:true}));
            //catches uncaught exceptions
            process.on('uncaughtException', exitHandler.bind(null, {exit:true}));

function saveTagToMysql(tag) {
    // Remove the "ok tag_id="-section from the tag.
    tag = tag.split("ok tag_id=")[1];
    // Remove the \r\n-newline from the tag.
    tag = tag.split("\r\n")[0];

    //  10. Create SQL-query for adding RFID-tags to the database.
    var sql_query = 'INSERT IGNORE INTO rfidtaulu (rfidkolumni) ' + 'VALUES("' + tag
+ '")';
    // Execute the query.
    if (debug_instead_of_mysql) {
        console.log("DEBUG: " + sql_query);
    } else {
        dbconnection.query(sql_query);
    }
}
// 11. Create a TCP client.
var client = new net.Socket();
// Connect to the Sirit-device, set the antenna power and request for a initial tag.
client.connect(device_port, device_ip, function() {
    console.log('Connected to the reader, reading tags every 0.5sec');
    // 12. Set the antenna power to the MAX!!!
    client.write(antenna_power_string);
    client.write(command_string);
});
client.on('data', function(data) {
    // 13. Query for the tag every 500 milliseconds.
    setTimeout(function() {
        var sirit_response = data.toString();
        if (sirit_response.match(/no_tag_in_field/)) {
            // Tag not found, resume looping for tags.
```

```
            client.write(command_string);
        } else if (sirit_response.match(/ok tag_id/)) {
            // 14. Tag found, save it to MySQL-table.
            saveTagToMysql(sirit_response);
            // 15. Request a new tag.
            client.write(command_string);
        }
    }, 500);
});
// 16. Closing
client.on('close', function() {
    console.log('Connection closed');
    process.exit(1);
});
```

Liite 8 Tietokannan sisältö



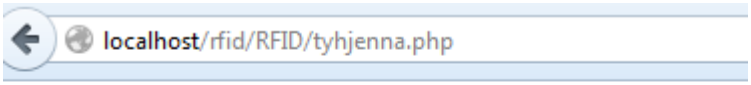
Tietokannan sisältösivu. Tietokannan sisältö esitetään taulukossa.

Lähdekoodi tunnistesivu.php

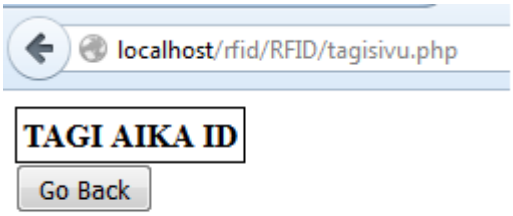
```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<?php
echo "<table style='border: solid 1px black;'>";
 echo "<tr><th>TAGI</th><th>AIKA</th><th>ID</th></th></tr>";
class TableRows extends RecursiveIteratorIterator {
    function __construct($it) {
        parent::__construct($it, self::LEAVES_ONLY);
    }
    function current() {
        return "<td style='width: 150px; border: 1px solid black;'>" . parent::current().
"</td>";
    }
    function beginChildren() {
        echo "<tr>";
    }
    function endChildren() {
        echo "</tr>" . "\n";
    }
}
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "nodejs";
try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $stmt = $conn->prepare("SELECT rfidkolumni, rfidtime, rfidid FROM rfidtaulu");
    $stmt->execute();
    // set the resulting array to associative
    $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
    foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as
$k=>$v) {
```

```
        echo $v;
    }
}
catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
}
$conn = null;
echo "</table>";
?>
 <button onclick="goBack()">Go Back</button>
<script>
function goBack() {
   window.history.back();
}
</script>
</body>
</html>
```

Liite 9 Tietokannan tyhjennys



Tietokannan tyhjennyssivu. Sivulle siirtyminen aiheuttaa tietokannan tyhjennyksen.

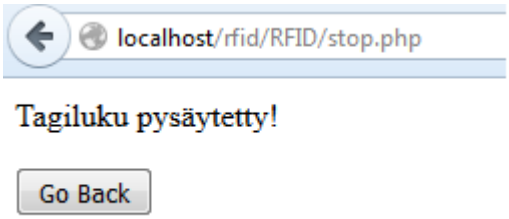


Tyhjä taulukko tyhjennyksen jälkeen.

Lähdekoodi tyhjenna.php

```
<?php
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "nodejs";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
// sql to delete a record
$sql = "truncate table rfidtaulu";
if ($conn->query($sql) === TRUE) {
   echo "Tietokanta tyhjennetty onnistuneesti!";
} else {
   echo "Virhe tietokannan tyhjennyksessä!: " . $conn->error;
}
$conn->close();
?>
<p></p>
 <button onclick="goBack()">Go Back</button>
<script>
function goBack() {
   window.history.back();
}
</script>
```

Liite 10 Lukijan pysäytys



Lukijan lukuprosessin pysäytyssivu. Sivulle siirtyminen pysäyttää komentokehotteen eli lukijaskriptikin pysähtyy.

Lähdekoodi stop.php

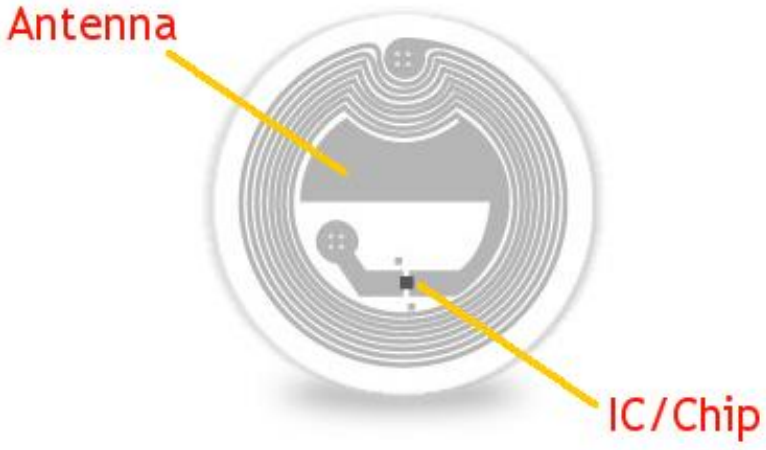
```
<!DOCTYPE html>
<html>
<body>
Tagiluku pysäytetty!
<p>
</p>
<?php
exec('c:\WINDOWS\system32\cmd.exe /c START
C:\xampp\htdocs\rfid\RFID\cmdclose.bat');
?>
 <button onclick="goBack()">Go Back</button>
<script>
function goBack() {
   window.history.back();
}
</script>
</body>
```

cmdclose.bat

```
taskkill /F /IM cmd.exe /T
```

Liite 11 NFC-lukija ja tunniste

Tavallinen NFC-pöytälukija. Tunniste viedään NFC-tekstin päälle, jolloin se tunnistetaan.



NFC-tunnisteen rakenne. Antenni kiertää rakenteen ympäri, jonka keskiössä on NFC-siru.

(NFC Reader. n.d.)